A Computational View on Natural Evolution

On the Rigorous Analysis of the Speed of Adaptation

Jorge Pérez Heredia

Department of Computer Science
University of Sheffield

This dissertation is submitted for the degree of
Doctor of Philosophy

January 2018

Declaration

I hereby declare that except where specific reference is made to the work of others, the contents of this dissertation are my own original work and have not been submitted in whole or in part for consideration for any other degree or qualification in this, or any other university. Some pieces of this thesis are based on articles that have been published elsewhere as specified in Section 1.1.

Jorge Pérez Heredia
January 2018

Acknowledgements

I would like to express my deep gratitude and appreciation to my supervisor Dr. Dirk Sudholt, I could not think of a better mentor. I would like to thank you for all your support, advice and for allowing me to grow as a research scientist. I am also greatly thankful to Tiago Paixão and Barbora Trubenová for being the "biology" side of my PhD, many thanks for all the fruitful discussions, insights and for hosting us at the Institute of Science and Technology (IST) several times. I would also like to specially thank Pietro Oliveto for his role as my second supervisor and coauthor in several papers.

For introducing me to interdisciplinary work and for the financial support to attend many research events, I would like to deeply thank all the Speed of Adaptation in Population Genetics and Evolutionary Computation (SAGE) project members and collaborators: Per Kristian Lehre, Tiago Paixão, Tobias Friedrich, Dirk Sudholt, Duc Cuong Dang, Golnaz Badkobeh, Andrew Sutton, Barbora Trubenová, Dogan Corus, Martin Krejca and Timo Kötzing. I specially thank Tobias Friedrich, Timo Kötzing and Martin Krejca for hosting me at the Hasso-Plattner-Institut (HPI).

I would also like to thank my PhD buddies from the Algorithms group: Alasdair Warwicker, Edgar Covantes Osuna and Donya Yazdani. I thank as well all the members of the department that contributed to a enjoyable working atmosphere, specially to José Miguel Rojas Siles, Ricardo Marxer, Abdullah Alsharif, Mathew Hall (Popeyes), Adam Poulston, Georg Struth, Victor Gomes, David Paterson and Thomas White.

I also gratefully acknowledge the Department of Computer Science of the University of Sheffield for the financial support and for giving me the opportunity to start my PhD.

Last but not least, my partner Laura and my family also deserve my deep gratitude for their endless support and patience during my PhD at Sheffield.

Abstract

Inspired by Darwin's ideas, Turing (1948) proposed an *evolutionary search* as an automated problem solving approach. Mimicking natural evolution, evolutionary algorithms evolve a set of solutions through the repeated application of the evolutionary operators (mutation, recombination and selection). Evolutionary algorithms belong to the family of black box algorithms which are general purpose optimisation tools. They are typically used when no good specific algorithm is known for the problem at hand and they have been reported to be surprisingly effective (Eiben and Smith, 2015; Sarker et al., 2002).

Interestingly, although evolutionary algorithms are heavily inspired by natural evolution, their study has deviated from the study of evolution by the population genetics community. We believe that this is a missed opportunity and that both fields can benefit from an interdisciplinary collaboration. The question of how long it takes for a natural population to evolve complex adaptations has fascinated researchers for decades. We will argue that this is an equivalent research question to the runtime analysis of algorithms.

By making use of the methods and techniques used in both fields, we will derive plenty of meaningful results for both communities, proving that this interdisciplinary approach is effective and relevant. We will apply the tools used in the theoretical analysis of evolutionary algorithms to quantify the complexity of adaptive walks on many landscapes, illustrating how the structure of the fitness landscape and the parameter conditions can impose limits to adaptation. Furthermore, as geneticists use diffusion theory to track the change in the allele frequencies of a population, we will develop a brand new model to analyse the dynamics of evolutionary algorithms. Our model, based on stochastic differential equations, will allow to describe not only the expected behaviour, but also to measure how much the process might deviate from that expectation.

Table of contents

Nomenclature **xiii**

I Introduction and Background 1

1 Introduction **3**

1.1 Underlying Publications 7

2 Evolutionary Algorithms **9**

2.1 Fitness Functions 12

2.2 Trajectory-Based Algorithms 15

2.2.1 Mutation Operator 16

2.2.2 Selection Operator 19

2.2.3 Popular Trajectory-Based Heuristics 20

2.3 Runtime Analysis of Evolutionary Algorithms 24

2.3.1 Markov Chains 27

2.3.2 Gambler's Ruin Problem 29

2.3.3 The Coupon Collector Problem 31

2.3.4 Drift Analysis 32

2.4 Fixed Budget 39

3 Population Genetics **43**

3.1 Selection, Mutation and Genetic Drift 44

3.1.1 Selection 45

3.1.2 Mutation 49

3.1.3 Genetic Drift 51

3.2 Diffusion Theory 51

3.2.1 Mutation, Selection and Genetic Drift under Diffusion Theory . . . 55

3.2.2 Probability of Fixation 57

3.2.3 The Strong Selection Weak Mutation Regime 59

II Runtime Analysis of a Natural Evolutionary Regime 63

4 Runtime Analysis and the Speed of Adaption 65
4.1 SSWM as a Trajectory-Based Algorithm . . . 69
4.2 Understanding the Fixation Probability . . . 71
4.3 Conclusions . . . 78

5 Speed of Adaptation in Additive Landscapes 79
5.1 Simple Hill Climbing Tasks . . . 80
5.2 Fitness Ridges . . . 88
5.3 Adaptation in a General Class of Landscapes . . . 90
5.4 Detecting the Steepest Slope . . . 92
5.5 Conclusions . . . 97

6 When Non-Elitism Outperforms Elitism for Crossing Fitness Valleys 99
6.1 Long Paths . . . 103
6.2 Crossing Simple Valleys . . . 105
6.2.1 Analysis for the (1+1) EA . . . 105
6.2.2 A General Framework for Local Search Algorithms . . . 106
6.2.3 Application to SSWM . . . 112
6.2.4 Application to the Metropolis algorithm . . . 116
6.3 Crossing Concatenated Valleys . . . 117
6.3.1 Application for SSWM and the Metropolis algorithm . . . 123
6.4 Global Mutations Speed-up Sharp Cliff Optimisation . . . 125
6.5 Conclusions . . . 132
Appendix 6.A Omitted Proofs from Subsection 6.2.2 . . . 133
Appendix 6.B Omitted Proofs from Subsection 6.2.4 . . . 134
Appendix 6.C Omitted Proofs from Section 6.4 . . . 137

7 When is it Beneficial to Reject Improvements? 141
7.1 A Common Stationary distribution . . . 143
7.2 A 3 State Model . . . 144
7.2.1 Experiments . . . 148
7.3 A 5 State Model . . . 149
7.3.1 An Example Where SSWM Outperforms the Metropolis algorithm . 153
7.3.2 Experiments . . . 157
7.4 When is it Beneficial to Exploit? . . . 159
7.4.1 Analysis for the 3 State Model . . . 160

7.4.2 Analysis for the 5 State Model 160
7.4.3 Experiments 164
7.5 Conclusions 165

III An Application of Stochastic Differential Equations to Evolutionary Algorithms 169

8 Modelling Evolutionary Algorithms with Stochastic Differential Equations 171
8.1 Stochastic Differential Equations 173
8.2 The Diffusion Approximation 175
8.2.1 Are Evolutionary Algorithms Diffusive Processes? 177
8.3 Drift Theorems for Fixed Budget Analysis 178
8.3.1 Additive Drift 178
8.3.2 Multiplicative Drift 179
8.3.3 Non-elitist Multiplicative Drift 181
8.4 Applications 183
8.4.1 Elitist Algorithms on LEADINGONES 183
8.4.2 Elitist Algorithms on ONEMAX 186
8.4.3 Non-Elitist Algorithms on ONEMAX 189
8.5 Validation of the Diffusion Approximation 191
8.5.1 Comparison with the Literature 191
8.5.2 Stationary Distribution 192
8.5.3 Reconciling Fixed Budget with Runtime Analysis 193
8.5.4 Simulations 197
8.5.5 Experimental Error 199
8.6 Conclusions 200

IV Conclusions and Outlook 203

9 Conclusions 205

References 209

Appendix A Probability Theory 221

Appendix B Diffusion Theory 223
B.1 Kolmogorov Forward Equation 223
B.2 Kolmogorov Backward Equation 224

Nomenclature

Mathematical Symbols

α	Reciprocal of the temperature T for the Metropolis Algorithm, $\alpha = 1/T$
β	Scaling parameter of the SSWM algorithm
Δf	Fitness difference between two bitstrings x and y, $\Delta f = f(y) - f(x)$
$\mathrm{E}[X]$	Expectation of the random variable X
$\mathrm{H}(x,y)$	Hamming distance between the bitstrings x and y, where $x,y \in \{0,1\}^n$
λ	Number of offspring created in one generation
$\ln(x)$	Natural logarithm of x
$\mathrm{Prob}(\mathrm{A})$	Probability of the event A
$\overline{f}$	Mean population fitness, see Definition 3.2
$\mathrm{mut}(x,y)$	Probability of mutation sampling state y from state x
$\mathbb{N}$	The set of the natural numbers, $\mathbb{N} = 1,2,3,\ldots$
$\lvert x \rvert_1$	Number of bits with value 1 in $x \in \{0,1\}^n$
$p_{\mathrm{acc}}(\Delta f)$	Probability of accepting a fitness change of Δf
$p_{\mathrm{fix}}(\Delta f)$	Acceptance probability of the SSWM algorithm, also known as the fixation probability
$\pi(x)$	Probability of sampling state x at equilibrium
$\mathbb{R}$	The set of the real numbers
$\mathbb{R}^+$	The set of the positive real numbers

$p^t_{x\to y}$	Transition probability of state x to state y in t iterations
$p_{x\to y}$	Transition probability of state x to state y in one iteration
$\mathrm{Var}[X]$	Variance of the random variable X
$\lvert x\rvert_0$	Number of bits with value 0 in $x \in \{0,1\}^n$
$\{0,1\}^k$	String of k elements from the set $\{0,1\}$, i.e., a bitstring
$\{X_t\}_{t\geq 0}$	An infinite time sequence of random variables $X_0, X_1, \dots$
e	Euler's number $e = \exp(1) = 2.7182\dots$
$f(x)$	Fitness of the solution x, $f : \{0,1\}^n \to \mathbb{R}$
$f = \Omega(g)$	Function f grows at least at fast as function g, see Definition 2.8
$f = O(g)$	Function f grows at most at fast as function g, see Definition 2.8
f_i^*	Marginal fitness, see Definition 3.2
f_{ij}	Genotypic fitness, see Definition 3.2
N	Population size of the SSWM algorithm
n	Problem size, number of bits of a solution.

Acronyms / Abbreviations

BILS	Best-Improvement Local Search
EA	Evolutionary Algorithm
EC	Evolutionary Computing
FILS	First-Improvement Local Search
GR	Gambler's Ruin
H-W	Hardy Weinberg
MA	Metropolis Algorithm
MAPE	Mean Absolute Percentage Error
PG	Population Genetics

RLS	Randomised Local Search
RMSE	Root Mean Squared Error
RW	Random Walk
SBM	Standard Bit Mutations
SDE	Stochastic Differential Equation
SSWM	Strong Selection Weak Mutation

Part I

Introduction and Background

Chapter 1

Introduction

Evolution has fascinated philosophers and scientists for more than 2000 years. The concept of *inheritance of acquired characteristics*, one of the first attempts to describe evolution, can be traced back to ancient Greek philosophers like Hippocrates (circa 400 BCE) or Aristotle (circa 300 BCE). This hypothesis states that an individual can transfer the physiological changes acquired during its lifetime to its offspring. However, it is fair to consider that the scientific study of evolution started in the early 1800s with Lamarck (1809). In his book *Philosophie Zoologique*, he coherently formulated for the first time the already known theory of inheritance of acquired characteristics.

Roughly fifty years later, Darwin (1859) and Wallace (1855) independently proved wrong Lamarck's proposed evolutionary mechanism. Although Darwin accepted Lamarckian inheritance, he claimed that mutations were random and the environmental factors played a role only on selection. On the other hand, Lamarck described evolution as a process directed by mutations which were caused by environmental factors.

After reading the work on population growth of the economist Malthus (1798), Darwin realised that while natural resources are limited, species are reproducing exponentially. Hence, individuals are in competition for resources and if the variations present in living organisms are hereditary, *nature will be selecting the fitter individuals*.

Through the repeated accumulation of beneficial small variations, evolution has managed to create highly adapted organisms that can survive under extreme conditions. Although it is well established that evolution does not progress towards a goal (see e.g. Barton et al., 2007), it shows similarities with one of the most important topics in Science and Technology: *problem optimisation*.

Problem optimisation is the search for the best choice, according to some criterion, among a collection of potential choices. It is a problem that appears in everyday life, but more importantly it arises in almost every scientific field: physical systems follow the principle of *minimal action*, engineers design motors that use as little fuel as possible, or banks aim to minimise investment risks while maximising the profit.

The human interest for optimisation also started with the ancient Greeks. They focused mainly on geometrical problems, as for example Euclid (circa 300 BCE) computed the minimal distance between a point and a line. Nevertheless, we have to wait until the late Middle-Age for a breakthrough in the field of optimisation. The foundations were set by Fermat (1636) showing that the derivative of a function vanishes at extreme points, along with Newton (1660s) and Leibniz (1670s) independently developing the *calculus of variations*. However, Fermat's method based in the analytic solution of the problem through the derivative, is (in general) only useful for didactic purposes. Real world problems might be described by a non-differentiable function, or we might not even know the function behind our optimisation task.

To circumvent this obstacle, scientists came up with *numerical methods*. Euler (1744) properly turned the calculus of variations into a research field and developed the famous method that carries his name. Half century later, the field had an extraordinary improvement with Gauss (1795) and Legendre (1806) independently presenting the *least square* method and Cauchy (1847) developing the *gradient method*. The key idea behind numerical methods is that a candidate solution is updated through the repeated application of a deterministic rule. These methods have proven to be highly successful optimising convex functions. However, since these algorithms typically use information of the function's slope in their update rule, they get stuck in extreme points and are not able to escape local optima.

In general, each problem has an inherent difficulty, some of them belong to the so-called NP complexity class. Although not proven, it is widely accepted that the time needed to find a solution for NP problems grows super polynomially with the size of the problem. With all hope lost to efficiently solve these problems, Barnard and Simon (1947) introduced the concept of a *heuristic*. Heuristics are strategies for problem solving which can efficiently find *sufficiently good* solutions, but they are not guaranteed to be optimal.

Following the heuristic approach and inspired by Darwin's ideas, Turing (1948) proposed an *evolutionary search* as an automated problem solving approach. Mimicking natural evolution, evolutionary algorithms (EAs) through the repeated application of variation operators (mutation, recombination and selection) evolve a set (population) of solutions (individuals). The strength of these algorithms relies on the stochasticity of the operators, which when well designed, will lead to an artificial evolution towards an optimal solution.

During the second half of the 20th century, this idea matured enough to become the research field today known as *Evolutionary Computing* (EC). In this period of time researchers developed a plethora of new methods which we can classify within four main branches:

- Evolutionary Programming (EP) by Fogel et al. (1965).
- Genetic Algorithms (GAs) by Holland (1973).
- Evolutionary Strategies (ES) by Rechenberg (1973) and Schwefel (1993).
- Genetic Programming (GP) by Koza (1992).

Evolutionary algorithms are general-purpose randomised heuristics that have been reported successful in countless occasions (Eiben and Smith, 2015; Sarker et al., 2002). No particular problem-knowledge is required for their application and yet they can be surprisingly effective, including some real-world problems (Luque and Alba, 2011). However, despite their versatility and efficiency to deliver satisfactory solutions, there is a lack in the theoretical understanding of these optimisers. The field does not posses a general mathematical theory from where arguments and hypothesis can be postulated to later be tested by experiments. As the reader will appreciate during this thesis, obtaining such general theory is an extremely hard task. In addition to the astronomically big number of algorithms and problems to consider, the stochastic nature of EAs makes their theoretical analysis highly involved. Nevertheless, in a short period of time, researchers have managed to develop a collection of powerful methods for the theoretical study of randomised search heuristics.

Interestingly, although evolutionary algorithms are inspired by natural evolution, the two research fields have deviated from each other. This is a missed opportunity, both fields can benefit from a interdisciplinary collaboration. Actually, we can find equivalent research questions: the time it takes for a natural population to reach a fitness peak is an important question for the study of natural evolution. Whereas in computer science, one of the most important questions, is the time needed for an algorithm to finish its execution.

Since the 1990's there has been a significant effort to establish the missed communication between biology and EC. Mühlenbein and Schlierkamp-Voosen (1993) got inspiration from human breeders to present the Breeder Genetic Algorithm (BGA) which they modelled using methods from *quantitative genetics*. Muhlenbein and Mahnig (2002) also established the validity of Wright's equation (see Equation (3.7)) for the Univariate Marginal Distribution Algorithm (UMDA) on a specific problem. Additionally, Mühlenbein (2009) studied a distributed evolutionary algorithm to back up what he called *Darwin's continent-island cycle conjecture*. In addition to Mühlenbein's work, recently there has been a renewed interest in applying computer science methods to problems in evolutionary biology with contributions from unlikely fields such as game theory (Chastain et al., 2014), machine learning (Valiant, 2009), Markov chain theory (Chatterjee et al., 2014) or the formal Darwinism project (Grafen, 2014).

However PG and EC are still far from being described by a unified theory. Noticing this research gap, a group of biologists and computer scientists formed the Speed of Adaptation in Population Genetics and Evolutionary Computation (SAGE) project, which was funded by the European Union Seventh Framework Programme (FP7/2007-2013). Among many other research achievements, the members of the project developed a unifying framework, which represents a serious attempt towards reconciling both communities (Paixão et al., 2015). With the same aim and as part of the SAGE project, this PhD thesis focuses on bridging Population Genetics (PG) and Evolutionary Computation. Our work has substantially contributed to breaking down the walls between both communities, exploiting the unexplored intersection between both fields. We will make use of the methods and techniques used in PG and EC to derive plenty of meaningful results for both communities, showing that this interdisciplinary approach is effective and relevant. Proof of this is that our work has been published in high rated conferences and top journals of both fields (see Section 1.1).

Firstly, we will apply the tools used in the theory of evolutionary algorithms to quantify the complexity of adaptive walks on many landscapes, illustrating how the structure of the fitness landscape and the parameter conditions can impose limits to adaptation. Secondly, EC will also be highly benefited from this synergetic relationship. Although the theoretical study of non-elitist algorithms is still in its early days, it is a well studied characteristic of natural evolution. As geneticists use diffusion theory to track the change in the allele frequencies of a population, we will develop a new model to analyse the dynamics of evolutionary algorithms. Our model, based on stochastic differential equations (SDEs), will allow to describe not only the expected behaviour, but also to measure how much the process might deviate from that expectation. This approach will yield the first fixed budget result for non-elitist algorithms.

After introducing the reader to the fields of Evolutionary Algorithms (Chapter 2) and Population Genetics (Chapter 3), we will analyse from a computational view, one of the most studied evolutionary regimes: the Strong Selection Weak Mutation (SSWM) regime. In Chapter 4 we will cast this process as an evolutionary algorithm, this will allow us to apply the runtime analysis techniques for randomised search heuristics (see Section 2.3). The analysis will yield interesting insights on the efficiency of this regime when adapting on different fitness landscapes. We will also investigate the parameters' values that makes this regime a good hill-climber (Chapter 5) or valley-crosser (Chapter 6).

Furthermore, we will compare the algorithmic version of this regime against some well-known evolutionary algorithms. Our results will allow us to answer questions of general significance: How to decide in advance which algorithm is preferable for valley crossing? (Chapter 6) Or when is it useful to reject improvements? (Chapter 7).

Finally in Chapter 8, we will consider the other missing direction of communication: applying the mathematical techniques used in the analysis of natural evolution to evolutionary computing. Particularly, we introduce the use of stochastic differential equations as a modelling technique for evolutionary algorithms. Building on top of the so-called *diffusion approximation* we formulate a hypothesis that we will use as a model for the dynamics of EAs. For some scenarios, we will be able to solve this equation analytically, and produce equivalent statements of two well-known drift theorems: *additive* and *multiplicative* drifts (see Subsection 2.3.4). The key idea in drift analysis is to focus on the expected progress of the stochastic process of interest (i.e., the drift). Typically, one tries to find easier expressions for the drift of the studied algorithm. This alternative drift has to be always slower (or faster) than the real drift. Then, we can derive pessimistic (or optimistic) estimations for the real runtime. Furthermore, we present a new more general multiplicative drift theorem for non-elitist algorithms. Finally, we perform a thorough validation of our hypothesis where we contrast our findings against the literature and experiments.

1.1 Underlying Publications

The contents of this thesis are based on the following publications. Authors' names are sorted alphabetically (except [2]).

Chapters 4 and 5 are based on the following papers:

1. Paixão, T., Pérez Heredia, J., Sudholt, D., and Trubenová, B. (2017). Towards a runtime comparison of natural and artificial evolution. *Algorithmica*, 78(2):681–713.

2. Pérez Heredia, J., Trubenová, B., Sudholt, D., and Paixão, T. (2017). Selection limits to adaptive walks on correlated landscapes. *Genetics*, 205(2):803–825, 2017.

   A preliminary version was published in:

   Paixão, T., Pérez Heredia, J., Sudholt, D., and Trubenová, B. (2015). First steps towards a runtime comparison of natural and artificial evolution. *Proceedings of the Genetic and Evolutionary Computation Conference 2015 (GECCO '15)*, pages 1455–1462. ACM.

Chapter 6 is based on the following paper:

3. Oliveto, P. S., Paixão, T., Pérez Heredia, J., Sudholt, D., and Trubenová, B. (2017). How to Escape Local Optima in Black Box Optimisation: When Non-Elitism Outperforms Elitism. In *Algorithmica*. To appear.

A preliminary version was published in:

Oliveto, P. S., Paixão, T., Pérez Heredia, J., Sudholt, D., and Trubenová, B. (2016) When non-elitism outperforms elitism for crossing fitness valleys. *Proceedings of the Genetic and Evolutionary Computation Conference 2016 (GECCO '16)*, pages 1163–1170. ACM.

Chapter 7 is based on the following paper:

4. Nallaperuma, S., Oliveto, P. S., Pérez Heredia, J., and Sudholt, D. (2017). On the Analysis of Trajectory-Based Search Algorithms: When is it Beneficial to Reject Improvements or to Exploit? Submitted to *Algorithmica*.

   A preliminary version was published in:

   Nallaperuma, S., Oliveto, P. S., Pérez Heredia, J., and Sudholt, D. (2017). When is it beneficial to reject improvements?. *Proceedings of the Genetic and Evolutionary Computation Conference 2017 (GECCO '17)*, pages 1455–1462. ACM.

Chapter 8 is based on the following paper:

5. Pérez Heredia, J. (2017). Modelling Evolutionary Algorithms with Stochastic Differential Equations. In *Evolutionary Computation*. To appear.

   A preliminary version was published in:

   Paixão, T. and Pérez Heredia, J. (2017). An application of stochastic differential equations to evolutionary algorithms. *Proceedings of the 14th ACM/SIGEVO Conference on Foundations of Genetic Algorithms, FOGA '17*, pages 3–11. ACM.

Chapter 2

Evolutionary Algorithms

Evolutionary algorithms are randomised search heuristics inspired by natural evolution. These algorithms simulate a population of individuals that through the repeated application of the *evolutionary operators* can evolve and adapt to the environment. The main goal of these heuristics is problem optimisation, i.e., given an objective function, known as *fitness function*, the algorithm's population will try to improve its fitness iteration by iteration. However, we can find other motivations for the study of evolutionary algorithms, for example Holland (1975) proposed EAs as a means to study adaptation.

EAs belong to the family of black box algorithms, they are general-purpose heuristics typically used when no good problem specific algorithm is known. Moreover, they are easy to implement and have been reported successful for many optimisation tasks where many exact techniques fail (Eiben and Smith, 2015), including real-world problems (Luque and Alba, 2011; Neumann and Witt, 2010). In order to improve the objective function's value, EAs rely on the repeated application of some stochastic operators. If these operators are well designed for the problem at hand, the population's mean fitness will increase, depicting an artificial evolutionary process (Eiben and Smith, 2015).

In this thesis, we will consider the maximisation of pseudo-Boolean functions $f : \{0,1\}^n \to \mathbb{R}$. This is the class of functions that assigns a real number to each point of the Boolean hyper-cube. Although we only consider maximisation, it is straightforward to notice that minimisation problems can be obtained just by multiplying the fitness function by -1. The search or *genotype space* $\{0,1\}^n$, together with the real and permutation space constitute the three most used search space representations (Jansen, 2013). We will not consider the real and permutation spaces in this thesis, however it is possible to construct both of them from the Boolean hyper-cube (with limited precision for the real space). Moreover, we can also establish a mapping between representing solutions as bit-strings and the DNA encoding present in living organisms (Paixão et al., 2015).

Although the behaviour of an EA varies with the choice of the evolutionary operators, we can observe the following standard underlying scheme. First, the algorithm initialises a population which is a multiset of candidate solutions (also known as individuals, search points or genotypes). Then, in each iteration until a termination criterion is met, two main phases are repeated: *variation* and *selection*. The variation phase refers to how new solutions are produced and is composed itself of another two phases: *recombination* and *mutation*. Finally, the selection phase decides which individuals survive to the next iteration.

Algorithm 2.1: Standard Evolutionary Algorithm

```
Initialise $\mu$ individuals
repeat
    Select a multiset of parents from the current population
    Create $\lambda$ new individuals by recombining the selected population
    Apply mutation to the offspring population
    From the union of the parent and offspring populations select $\mu$ individuals
until termination condition;
```

Each of the steps outlined in the above pseudo-code can be altered or adapted at the pleasure of the programmer. However, it is important to keep in mind that variations of these operators typically yield a different algorithmic behaviour. In some scenarios even the slightest change can yield a huge performance drop (Jansen, 2007).

Initialisation

A smart initialisation mechanism can significantly improve the performance of the algorithm by starting in the correct area of the search space (see e.g. Friedrich et al., 2017). However, as we mentioned earlier, EAs are usually applied when no much a priori problem knowledge is known, hence initialisation is typically kept simple. Actually one of the most used mechanisms is *uniform initialisation*, i.e., selecting bit-strings uniformly at random.

Selection

There are many choices both for the parent and survival selection (code-lines 3 and 6 in Algorithm 2.1). For example, *uniform selection* is the simple case where individuals are selected uniformly at random. However, the original purpose of selection is to mimic the principle of *survival of the fittest*, hence it is reasonable that individuals with higher fitness have a higher probability of being selected, both for mating and for surviving to the next generation. Following this principle, we can find popular selection mechanisms such as:

fitness-proportional selection where the selecting probability is proportional to the fitness, or the extreme *cut selection* where a set of individuals with high fitness is deterministically chosen. In all the above cases, selection acts at the phenotype level, therefore it is independent on the search space representation.

It is not completely clear when one selection mechanism is preferred over another for a given problem. We will shed some light on this question by introducing in more detail some popular selection mechanisms (Subsection 2.2.2), and studying their influence when optimising different fitness landscapes (Parts II and III). We will find examples when non-elitism outperforms elitism and we will investigate the usefulness of rejecting improvements.

Recombination

The recombination or *crossover* operator is a *variation operator*. Unlike selection, this operator typically acts over the genotype space and hence is dependent on its representation. Again, we find a large variety of crossover mechanisms, some of them highly artificial, however one of the most used choices is the natural extension of sex in biological populations, the *uniform crossover*. Uniform crossover, as many recombination mechanisms, takes two individuals as input (binary operator $\mathrm{C} : \{0,1\}^n, \{0,1\}^n \to \{0,1\}^n$) and copies each bit randomly (and independently) from one of the parents.

The main motivation for the use of recombination is simple: when two parents possess different beneficial properties, we would like that the offspring inherits both. However, the same rationale works for detrimental properties. In the case of the continuous domain (which we do not cover in this thesis) crossover acts as a repairing mechanism (Beyer, 2001). However, for the Boolean hypercube representation, the research question is not settled. Nevertheless, crossover has been proved to be useful in EC for some scenarios (see Dang et al., 2017; Jansen and Wegener, 2002). Interestingly, we can find the same issues raised in Biology where the role of sex remains an open question (Barton and Charlesworth, 1998).

Mutation

Mutation is a variation operator that only uses one individual as input $\mathrm{M} : \{0,1\}^n \to \{0,1\}^n$ (unary operator). Typically, mutation uses only the genotypic information of the parent to produce a slightly different genotype. This implies that, as recombination, it will be dependent on the search space representation. In Subsection 2.2.1, we will study two of the most used mutation operators for the Boolean hyper-cube: *standard bit mutations* (SBM) and *k-bit mutations*. The former flips each bit independently with some probability (typically $1/n$), and the latter flips k-bits (typically $k = 1$) chosen uniformly at random.

Termination Criterion

Finally, we have to decide a termination criterion. We find two popular termination criteria namely *fixed* and *adaptive*. The former, as the name suggests, will let the algorithm run for a predetermined amount of time. On the other hand, an adaptive criterion will take into account information of the current status of the simulation and stop when some predetermined condition is met. Typical choices for adaptive termination criteria are when some satisfying fitness value is reached or when there is a lack of improvement.

However, in the theoretical study of EAs we disregard any stopping criteria and let the algorithm run forever. As we will explain in Section 2.3 we are interested in the first point in time when the algorithm reaches a specific fitness values (typically the optimum).

2.1 Fitness Functions

As mentioned in the previous section, pseudo-Boolean fitness functions map the Boolean hyper-cube into the real numbers $f : \{0,1\}^n \rightarrow \mathbb{R}$. Then, it is highly relevant to study the structure of the hyper-cube. Figure 2.1 shows the 2-dimensional projection of the hyper-cube, where the position of each bit-string is determined by: the number of 1-bits (vertical position) and the number of 1-bits until the first appearance of a 0-bit (horizontal position).

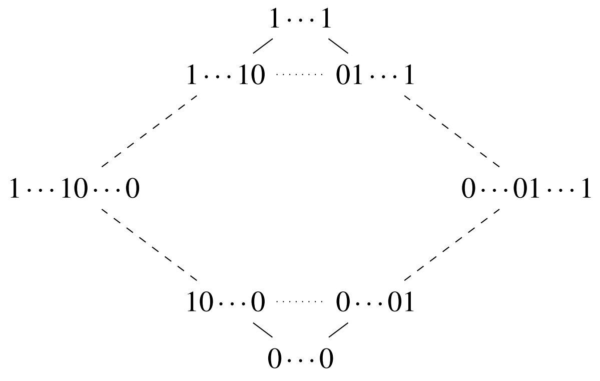


Fig. 2.1 Projection of the Boolean hyper-cube in two dimensions.

To study pseudo-Boolean functions, it seems natural to establish some metric for their search space. The most natural metric for the Boolean hyper-cube is the *Hamming distance*, which simply counts the number of mismatched bits between two bit-strings. If the number of different bits between two search points is just one, we will call them *Hamming neighbours*.

Definition 2.1 (Hamming distance). *Given two bit-strings $x,y \in \{0,1\}^n$, the Hamming distance between them is given by $H(x,y) := \sum_{i=1}^{n} |x_i - y_i|$, if $H(x,y) = 1$ we say that x and y are Hamming neighbours.*

Once a metric is defined, we can introduce the concept of local optima. We will denote by *local optima* those bit-strings without Hamming neighbours of higher fitness. And the *global optimum* will be the search point with the highest fitness, which also corresponds to the local optimum with highest fitness.

Definition 2.2 (Local and global optima)**.** *Given a function* $f : \{0,1\}^n \rightarrow \mathbb{R}$*, the set* $\mathbb{Y}$ *of local optima is given by* $\mathbb{Y} := \{y \mid x, y \in \{0,1\}^n,\ H(x,y) = 1,\ f(y) \geq f(x)\}$*. And the global optima* x_{opt} *are defined as* $x_{\text{opt}} := \operatorname{argmax}\{f(y) \mid y \in \{0,1\}^n\}$.

To describe the features of fitness functions we can use a metaphor with natural landscapes. A *fitness landscape* can be obtained by imagining that each search point is as high as its fitness. Then we can identify many similar features as in natural landscapes. The neighbourhood of fitness optima can resemble a *hill* (Figure 2.2, upper left); also we can observe *fitness valleys* – those regions surrounded by search points of higher fitness (Figure 2.2, upper right). Moreover, we can talk of the *basin of attraction* of an optimum as the regions of the search space where the fitness points towards such optimum (red zones in Figure 2.2). When a basin of attraction has a longitudinal shape depicting a *path* we talk about *fitness ridges*. Here, neighbouring points outside the path have lower fitness (Figure 2.2, bottom left). Finally, we can also find regions of constant fitness representing *plateaus* (Figure 2.2, bottom right).

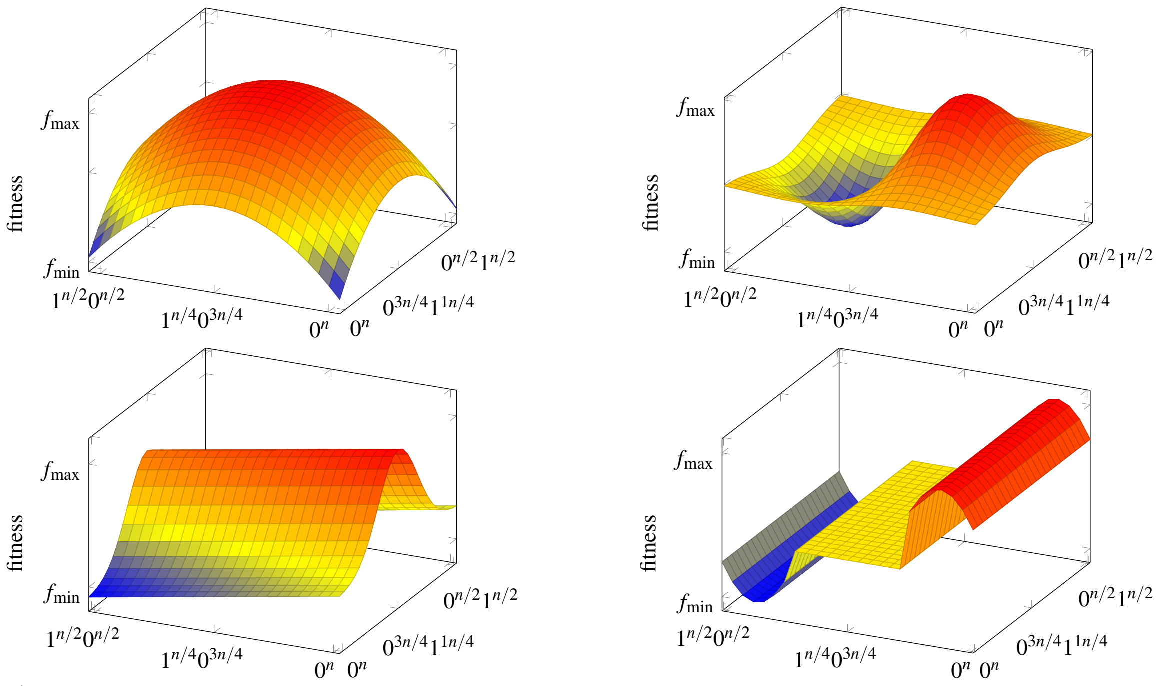


Fig. 2.2 Graphical representations of a fitness hill (upper left), a fitness valley (upper right), a fitness ridge (bottom left) and a fitness plateau (upper right). The horizontal plane corresponds with the Boolean hyper-cube from Figure 2.1. This figure constitutes an artistic representation, no mathematical projection was used to cast the Boolean hypercube into the horizontal plane.

Due to the high technical challenge to rigorously analyse randomised algorithms, researches usually use *toy problems*. These benchmark problems usually represent one or more landscape features which are believed to be building blocks for more complex functions. This way, we can derive initial analysis tools that can pave the way for further studies on more complex functions (Oliveto and Yao, 2011).

Another typical approach to theoretically study an algorithm is to design fitness landscapes where the algorithm behaves differently. Understanding on which situations an algorithm fails or succeeds is an important question which helps designing EAs (Corus et al., 2017). This way we learn about problem characteristics that lead to a good or bad performance for the considered algorithm. Furthermore, optimisation practitioners can highly benefit from this kind of theoretical results since they discard or suggest the application of a certain algorithm for the problem at hand.

In Parts II and III we will define and analyse many problems, but for now, we only introduce the two most theoretically studied toy problems: ONEMAX and LEADINGONES. The GENERALISEDONEMAX problem simply counts the number of mismatched bits between a given target solution and the current solution. Using the landscape metaphor, it represents an easy hill climbing task since each bit-position's fitness contains information about the direction towards the optimum.

Definition 2.3 (GENERALISEDONEMAX)**.** *Let $x, x_{\text{opt}} \in \{0,1\}^n$, then*

$$\text{GENERALISEDONEMAX}(x_{\text{opt}}, x) = n - H(x_{\text{opt}}, x).$$

Is it important to note that since the studied algorithms are unbiased (i.e., they do not favour flipping 0-bits into 1-bits or vice versa), the choice of the target solution x_{opt} does not affect the optimisation process. Hence, we can highly simplify the theoretical analysis by predefining our target. The typical choice for the target solution is the all ones bitstring, this way we can introduce a function that simply counts the number of ones: $\text{ONEMAX}(x) = \text{GENERALISEDONEMAX}(1^n, x)$.

Definition 2.4 (ONEMAX)**.** *Let $x \in \{0,1\}^n$, then*

$$\text{ONEMAX}(x) = \sum_{i=1}^{n} x_i.$$

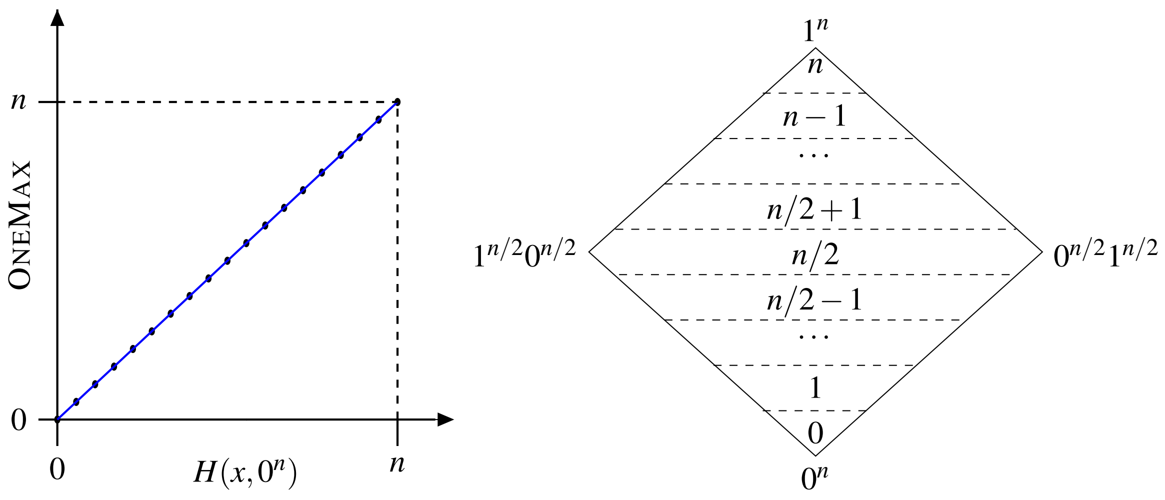


Fig. 2.3 Phenotype (left) and genotype (right) representation of the ONEMAX problem The layout for the genotype representation corresponds with the Boolean hypercube from Figure 2.1.

LEADINGONES is a function that counts the number of 1-bits at the beginning of the bitstring, i.e., until the appearance of the first 0-bit. Using again the landscape metaphor, this problem represents a fitness ridge since bit matches with the optimum do not always increase the fitness and losing one bit-match can yield a huge fitness loss. As in the previous case, the choice of the all ones bitstring as target solution does not affect the optimisation process.

Definition 2.5 (LEADINGONES). *Let $x \in \{0,1\}^n$, then*

$$\text{LEADINGONES}(x) = \sum_{i=1}^{n} \prod_{j=1}^{i} x_j.$$

2.2 Trajectory-Based Algorithms

Trajectory-based algorithms are search heuristics that evolve a single lineage rather than using a population. Alternatively, they can be seen as the extreme case of a standard EA with a population of size 1 (i.e. $\mu = 1$ in Algorithm 2.1). Trajectory-based algorithms obtain their name from the fact that they produce a sequence of solutions that corresponds with one trajectory in the search space.

Although they might seem simple, their theoretical study is highly relevant. Since their analysis is, in general, easier than analysing population-based algorithms, it is a good starting point that can pave the way for the analysis of more elaborated heuristics. Moreover, it can also yield insight on the usefulness of new analysis methods (Droste et al., 2002).

Within this family of heuristics we can find well-known algorithms such as: randomised local search (RLS), the Metropolis algorithm (MA), and simple EAs such as the (1+1) EA or the (1,λ) EA. In general, we can cast any trajectory-based algorithms with the following pseudo-code.

Algorithm 2.2: Trajectory-Based Algorithm

```
1 Initialise $x \in \{0,1\}^n$
2 repeat
3     $y \leftarrow$ MUTATE($x$)
4     $x \leftarrow$ SELECT($x,y$)
5 until stop;
```

As it can be observed in the above pseudo-code, the two crucial operators for trajectory-based algorithms are mutation and selection. Hence, we proceed to study these two mechanism in more detail.

2.2.1 Mutation Operator

As outlined in the previous section, there are many mutation mechanisms. However, we will consider the two most used mutation operators for trajectory-based algorithms:

- **Local mutations (a.k.a. 1-bit mutations):** Flip one uniform randomly chosen bit from the parent genotype.
- **Global mutations (a.k.a. bit-wise or standard bit mutations):** Flip uniformly at random each bit of the parent genotype with probability $1/n$.

For a better understanding, we graphically represent these two mutation operators for the 3-dimensional Boolean hyper-cube (Figure 2.4). We can observe how, local mutations can only produce search points that differ exactly in one bit with respect to the parent genotype (Hamming neighbours). Whereas global mutations provides a non-zero probability of moving to any point from the search space. However, we can observe that this probability decreases with the number of mismatched bits between the parent and the child genotype.

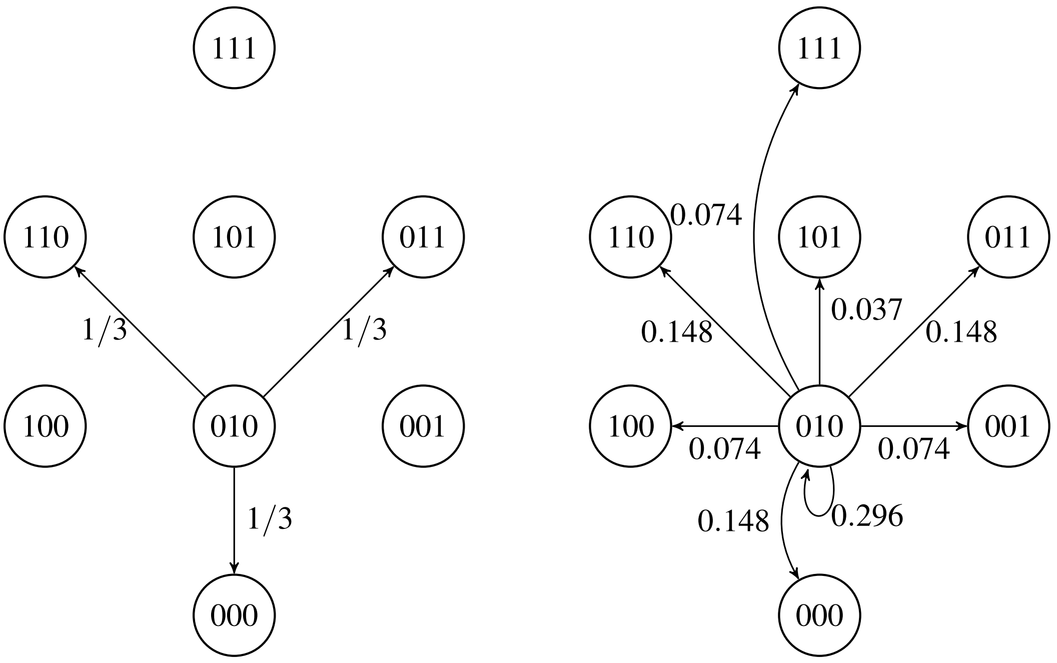


Fig. 2.4 Local (left) and global (right) mutation probabilities from the state 010 on the Boolean hyper-cube of dimension 3.

Properties of Standard Bit Mutations

The theoretical study of local mutations is fairly easy, however studying global mutations is more involved. In this subsection we present some useful properties of SBM mutations. An interesting feature of this mutation operator is that it can create any search point. We will see in Section 2.3 that this property is very important since it guarantees that the algorithm will converge in finite time, no matter the fitness function.

It is important however, to notice how the probability of creating each search point is distributed. In Figure 2.4 we can observe that, different points with the same Hamming distance from the parent genotype (bit-string 010) have the same probability of being generated. This is expected since, in order to create a search point that differs in k bits from the parent, it is necessary to flip those k bits (probability $1/n^k$) and not flip the remaining $n-k$ bits (probability $(1-1/n)^{n-k}$).

Definition 2.6 (Global Mutations)**.** *Given two bit-strings $x, y \in \{0,1\}^n$ with $H(x,y) = k$. The probability of global mutations sampling y from x, namely* $\mathrm{mut}(x,y)$*, is given by*

$$\mathrm{mut}(x,y) = \left(\frac{1}{n}\right)^k \cdot \left(1 - \frac{1}{n}\right)^{n-k}.$$

It is straightforward to notice that this probability is shared by all the search points which are at a Hamming distance k of x. Since there are $\binom{n}{k}$ bit-strings at a distance k from x, we can compute the probability of creating any such search point as $\binom{n}{k} \cdot \text{mut}(x,y)$. As sanity check, one can take the sum over all the possible values of k, to verify that no events were excluded $\sum_{k=0}^{n} \binom{n}{k} \cdot \text{mut}(x,y) = 1$. Finally, the following graph shows how the probability of global mutations creating a search point at Hamming distance of k from the parent genotype, exponentially decreases with such distance k.

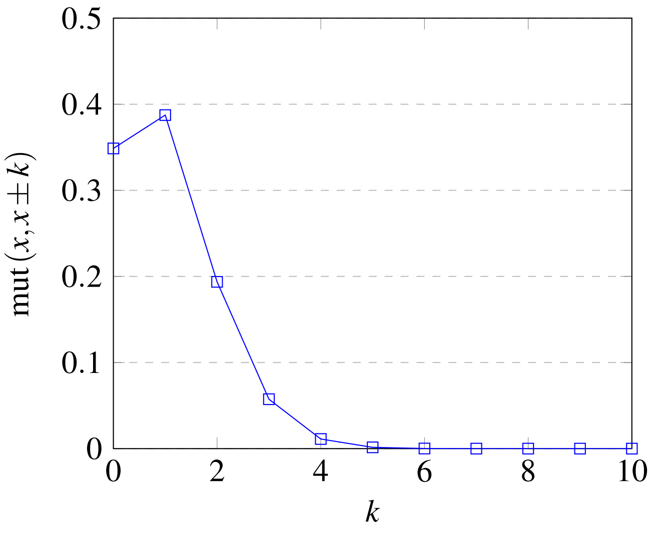


Fig. 2.5 Probability of SBMs creating a search point at a Hamming distance of k from the parent genotype x. Problem size $n = 10$.

As it will become obvious along this thesis, it is technically hard to maintain exact mathematical expressions when analysing global mutations. Instead, we will be using bounds as those derived in the following lemma.

Lemma 2.1. *[Lemma 3 in Paixão et al. (2017)] For any positive integer $k > 0$ and $0 \le x \le n$, let $\text{mut}(x, x \pm k)$ be the probability that a global mutation of a search point with x ones creates an offspring with $x \pm k$ ones. Then,*

$$\begin{aligned}
\text{mut}(x, x+k) &\le \left(\frac{n-x}{n}\right)^k \left(1 - \frac{1}{n}\right)^{n-k} \cdot \frac{1.14}{k!} \\
\text{mut}(x, x-k) &\le \left(\frac{x}{n}\right)^k \left(1 - \frac{1}{n}\right)^{n-k} \cdot \frac{1.14}{k!} \\
\text{mut}(x, x-k) &\le \frac{\text{mut}(x, x-1)}{k!} \le \frac{1}{k!}.
\end{aligned}$$

2.2.2 Selection Operator

As depicted in Algorithm 2.2, selection is the other main evolutionary operator for trajectory-based algorithms. In this thesis, we mainly study selection operators that can be specified by an acceptance probability p_{acc}. Here, a probability distribution decides if the new genotype produced by mutation is accepted or rejected; $p_{\text{acc}} : \mathbb{R} \to [0,1]$. Furthermore, we will consider algorithms where this acceptance probability depends on the fitness difference between the parent and offspring $\Delta f = f_{\text{child}} - f_{\text{parent}}$.

Algorithm 2.3: Trajectory-Based Algorithm

```
1 Initialise $x \in \{0,1\}^n$
2 **repeat**
3     $y \leftarrow$ MUTATE($x$)
4     $\Delta f \leftarrow f(y) - f(x)$
5     Choose $r \in [0,1]$ uniformly at random
6     **if** $r \leq p_{\text{acc}}(\Delta f)$ **then**
7         $x \leftarrow y$
8 **until** *stop*;
```

Within this class of algorithms, we can recover well-known heuristics depending on the acceptance probability. On one extreme, we find the absence of selection from a Random Walk (RW) process, which will always accept the new candidate move no matter its fitness.

$$p_{\text{acc}}^{\text{RW}}(\Delta f) = 1 \quad \forall\, \Delta f. \tag{2.1}$$

On the other side of the spectrum, we find elitist algorithms such as Randomised Local Search or the (1+1) EA. Here, the acceptance probability is the step function with threshold $\Delta f = 0$. Therefore, if mutation produces a point of lower fitness than the parent, it will be rejected. However, if the new search point is at least as good as the parent it will be accepted. These heuristics are typically referred to as *randomised hill climbers*.

$$p_{\text{acc}}^{\text{RLS}}(\Delta f) = p_{\text{acc}}^{\text{EA}}(\Delta f) = \begin{cases} 1 & \text{if } \Delta f \geq 0 \\ 0 & \text{if } \Delta f < 0. \end{cases} \tag{2.2}$$

All the other trajectory-based algorithms that can be described by an acceptance probability, will be in between these two limit cases. For example, if worsening moves are allowed with an exponentially decreasing probability but elitism is kept for improving moves, we find the so-called Metropolis algorithm (Metropolis et al., 1953).

$$p_{\text{acc}}^{\text{MA}}(\Delta f, \alpha \in \mathbb{R}^+) = \begin{cases} 1 & \text{if } \Delta f \geq 0 \\ e^{\alpha \Delta f} & \text{if } \Delta f < 0. \end{cases} \tag{2.3}$$

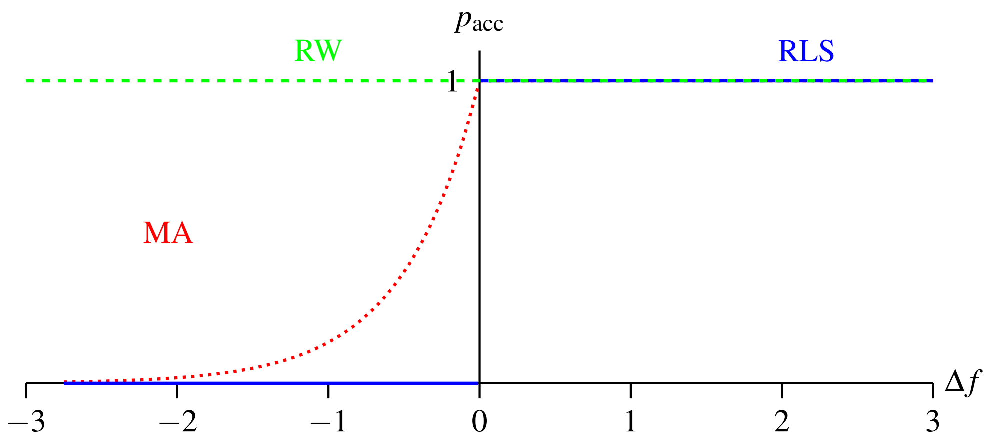


Fig. 2.6 Acceptance probability for RW (green dashed line), RLS and (1+1) EA (blue solid line) and the Metropolis algorithm (red dotted line). The horizontal axis represents the fitness difference between the child and the parent genotype $\Delta f = f_{\text{child}} - f_{\text{parent}}$.

2.2.3 Popular Trajectory-Based Heuristics

This subsection contains a compilation of the algorithms that we will be studying in this thesis. First, if we choose local mutations and the acceptance probability from Equation (2.1), Algorithm 2.3 becomes the well-known random walk.

Algorithm 2.4: Random Walk

```
1 Initialise $x \in \{0,1\}^n$
2 repeat
3     $x \leftarrow$ flip a uniform randomly chosen bit from $x$
4 until stop;
```

Keeping local mutations but using the acceptance probability from Equation (2.2) leads to the RLS algorithm.

Algorithm 2.5: Randomised Local Search

```
1 Initialise $x \in \{0,1\}^n$
2 repeat
3     $y \leftarrow$ flip a uniform randomly chosen bit from $x$
4     if $f(y) \geq f(x)$ then
5         $x \leftarrow y$
6 until stop;
```

Since RLS only produces Hamming neighbours and rejects those points of lower fitness, it gets stuck in local optima. One strategy to overcome this obstacle, while maintaining elitism, is to change the mutation operator to global mutations (Definition 2.6). In this case, we recover the most theoretically studied evolutionary algorithm, the (1+1) EA.

Algorithm 2.6: (1+1) EA

```
1 Initialise $x \in \{0,1\}^n$
2 repeat
3     $y \leftarrow$ flip each bit of $x$ uniform at random with probability $1/n$
4     if $f(y) \geq f(x)$ then
5         $x \leftarrow y$
6 until stop;
```

Another famous nature-inspired heuristic is the Metropolis algorithm (Metropolis et al., 1953). It is inspired by the so-called *Boltzmann distribution* which classical physical systems follow at thermal equilibrium. Here, the probability of finding a particle in a state of energy E is proportional to $e^{-E/(k_B T)}$, where T is the temperature and k_B is Boltzmann's constant. From an algorithmic perspective, the energy relates to the fitness and the temperature is just a parameter (typically its reciprocal is used; $\alpha = 1/T$).

Algorithm 2.7: Metropolis Algorithm

```
1  Initialise $x \in \{0,1\}^n$
2  Choose a temperature TEMP $> 0$
3  $\alpha \leftarrow 1/\text{TEMP}$
4  repeat
5      $y \leftarrow$ flip a uniform randomly chosen bit from $x$
6      $\Delta f \leftarrow f(y) - f(x)$
7      Choose $r \in [0,1]$ uniformly at random
8      if $r \leq p_{\text{acc}}^{\text{MA}}(\Delta f, \alpha)$ then
9          $x \leftarrow y$
10 until stop;
```

Another popular non-elitist trajectory-based algorithm is the (1,λ) RLS. This optimiser produces λ children by local mutations and selects the best one for survival (ties broken uniformly at random). Despite the selection mechanism being elitist, the (1,λ) RLS is not an elitist algorithm. Because the parent genotype was left out the fitness comparison, when the λ children have a lower fitness than the current solution, the algorithm will accept a move towards a worse fitness point (even if $\Delta f = -\infty$).

Algorithm 2.8: (1,λ) RLS

```
1  Initialise $x \in \{0,1\}^n$
2  Choose a population size $\lambda \geq 1$
3  repeat
4      for $i = 1$ to $\lambda$ do
5          $y_i \leftarrow$ flip a uniform randomly chosen bit from $x$
6      $x \leftarrow$ uniform randomly chosen from $\arg\max(f(y_1), f(y_2), \ldots, f(y_\lambda))$
7  until stop;
```

Finally, we will also consider the popular First-Improvement Local Search (FILS) and Best-Improvement Local Search (BILS) algorithms (see e.g. Wei and Dinneen, 2014). These two optimisers, like any Algorithm 2.2 with local mutations, can only explore the Hamming neighbourhood in one iteration. Whilst FILS will keep producing Hamming neighbours until it finds an improvement, BILS computes the set of all neighbours and chooses one of those with the highest fitness. Both algorithms stop when there is no improving neighbour.

Algorithm 2.9: FILS (Adapted from Algorithm 4 in Wei and Dinneen, 2014)

```
Initialise $x \in \{0,1\}^n$
$i \leftarrow 0$
repeat
    Generate a random permutation Per of length $n$
    for $i = 1$ to $n$ do
        $y \leftarrow$ flip the Per[$i$]-th bit of $x$
        if $f(y) > f(x)$ then
            $x \leftarrow y$
            go to line 4
    stop
until stop;
```

Algorithm 2.10: BILS (Adapted from Algorithm 3 in Wei and Dinneen, 2014)

```
Initialise $x \in \{0,1\}^n$;
repeat
    BestNeighbourSet $= \emptyset$
    for $i = 1$ to $n$ do
        $y \leftarrow$ flip the $i$-th bit of $x$
        if $f(y) > f(x)$ then
            BestNeighbourSet $\leftarrow$ BestNeighbourSet $\cup\ y$
    if BestNeighbourSet $= \emptyset$ then
        stop
    $x$ is uniform randomly chosen from $\arg\max$ (BestNeighbourSet)
until stop;
```

2.3 Runtime Analysis of Evolutionary Algorithms

Although the study of evolutionary algorithms is mainly experimental, we can trace back theoretical analyses to the seventies. The first theoretical attempts were aiming to explain the behaviour of EAs rather than analysing its performance. The most known of these approaches was the *Schema theory* by Holland (1975). Later, *Markov Chain* theory became the preferred tool (He and Yao, 2003; Vose, 1995). We can find further modelling attempts via fixed point analysis of Markov Chains by Vose (1995), and Wright and Rowe (2001) with a similar approach based on dynamical systems. But more importantly, the use of Markov Chains opened the doors to a new analysing perspective, *performance*.

The analysis of deterministic algorithms is mainly composed of two research questions: *correctness* and *runtime* (see e.g. Cormen et al., 2001). In other words, when a new algorithm is presented, we would like to be guaranteed that the algorithm will always deliver the correct answer, no matter the input. Secondly, we would like to know how much time the algorithm needs to achieve that answer. In the case of optimisation algorithms, analysing the correctness of the algorithm becomes analysing the convergence to the optimum. However, EAs are randomised algorithms and therefore, they are not suitable to be analysed with the classical notions of deterministic convergence. The question here is to prove that an EA finds the optimum of a specific function in finite time with probability one.

Fortunately, by using Markov Chains, Rudolph (1998) defined general conditions that made this question trivial thenceforth. This achievement allowed to move all the research effort towards runtime analysis. The key idea is that, if a randomised algorithm is described by an ergodic Markov Chain, i.e., all states are accessible at any time. Then, there is a non-zero probability $\Pr\left(x, x_{\text{opt}}\right)$ of transiting to the optimum state x_{opt} from any other state x. Since the number of trials needed for such an event to occur follows a geometric distribution, the expected waiting time will be $1/\Pr\left(x, x_{\text{opt}}\right)$ (see lemma A.3 in the Appendix A). The following theorem (adapted from Droste et al., 2002) presents a case study using this idea.

Theorem 2.1. *The expected time of the (1+1) EA to find the optimum of any pseudo-Boolean function is at most n^n iterations.*

Proof. Simply recalling that the (1+1) EA (Algorithm 2.6) uses global mutations (see definition 2.6) we can observe that: from any solution x, the (1+1) EA will produce the optimal solution x_{opt} with probability

$$\operatorname{mut}\left(x, x_{\text{opt}}\right) = \left(\frac{1}{n}\right)^k \cdot \left(1 - \frac{1}{n}\right)^{n-k} \geq \left(\frac{1}{n}\right)^n .$$

Where we have pessimistically assumed that the Hamming distance k between x and x_{opt} is the problem size n. Since the (1+1) EA will always accept a fitness improving move, the probability of moving to the optimum $\Pr\left(x, x_{\mathrm{opt}}\right) = \mathrm{mut}\left(x, x_{\mathrm{opt}}\right)$. Finally, we notice that the number of trials needed for such event to occur follows a geometric distribution, which leads an expected time of $1/\mathrm{mut}\left(x, x_{\mathrm{opt}}\right) \leq n^n$. $\square$

With the convergence question settled, researches focused on building methods for the runtime analysis of randomised search heuristics. Here, the research question is to derive a mathematical expression for the time T needed for an algorithm to finish its execution.

Definition 2.7 (Optimisation Time). *Let $\{X_t\}_{t\geq 0}$ be a stochastic process on the state space $\Omega = \{0,1\}^n$ and $f : \Omega \to \mathbb{R}$ a fitness function. The optimisation time T is the first point in time when the process' value yields the maximum fitness value, i.e.,*

$$T := \min\{t \geq 0 \mid X_t = \arg\max_{X_t \in \Omega} f(X_t)\}.$$

Although this is the current main research line in the field, there is an obvious hurdle: How can we know if the algorithm has seen an optimum?. As described in Section 2.1, we will only consider *toy problems* which are believed to be building blocks for more complex functions. For these problems, unlike in many real world problems, we know the optimal solution a priori, evading the hurdle. However, for many problems (e.g. travelling salesman) we cannot be sure that an optimum has been sampled. We will see in Section 2.4 how this is solved by the new perspective of *fixed budget* which looks at the fitnesses that have been observed. At the end of this thesis (Subsection 8.5.3) we will discuss further how to reconcile runtime analysis with fixed budget.

As mentioned before, EAs are randomised algorithms, hence the question translates to finding the expected time $\mathrm{E}[T]$ needed to find the optimal solution. This time is typically expressed with asymptotic notation in terms of the problem size n (Cormen et al., 2001).

Definition 2.8 (Asymptotic Notation). *For any two functions $f, g : \mathbb{N}_0 \to \mathbb{R}$, we say that:*

- *$f = O(g)$ if and only if there exist constants $c \in \mathbb{R}^+$ and $n_0 \in \mathbb{N}$, such that for all $n \geq n_0$ it holds that $f(n) \leq cg(n)$.*
- *$f = \Omega(g)$ if and only if $g = O(f)$.*
- *$f = \Theta(g)$ if and only if $f = O(g)$ and $f = \Omega(g)$.*
- *$f = o(g)$ if and only if $\lim_{n\to\infty} f(n)/g(n) = 0$.*
- *$f = \omega(g)$ if and only if $g = o(f)$.*

By using asymptotic notation we can express runtime results with a rigorous mathematical expression. Furthermore, we can establish different orders of growth as stated in the following definition. This way, we can establish efficiency criteria for evolutionary algorithms. Generally speaking, we will say that an algorithm with a superpolynomial (or higher growing order) optimisation time is inefficient for the problem at hand. In contrast to a polynomial runtime that denotes an efficient algorithm.

Definition 2.9. *For a function $f : \mathbb{N} \to \mathbb{R}^+$, we say that*

- *f is polynomial $f = poly(n)$ if $f(n) = O(n^c)$ for some constant $c \in R_0^+$.*
- *f is superpolynomial if $f(n) = \omega(n^c)$ for every constant $c \in \mathbb{R}_0^+$*
- *f is exponential if $f(n) = \Omega\left(2^{n^\varepsilon}\right)$ for some constant $\varepsilon \in \mathbb{R}^+$.*
- *f is polynomially small $f = 1/poly(n)$ if $1/f$ is polynomial.*
- *f is superpolynomially small if $1/f$ is superpolynomial.*
- *f is exponentially small if $1/f$ is exponential.*

But a mathematical expression and an order of growth for the expected optimisation time does not completely settle the question. Despite being able to guarantee that EAs find the optimum in finite time, we can not ensure that for a given amount of time an EA obtains the optimal solution with probability one (see e.g. Doerr et al., 2013). Furthermore, the distribution followed by the optimisation time T might be very disperse, and its expectation will not give useful information about the runtime. To mitigate these issues, we can usually find that runtime analysis results are presented in the form of an expected optimisation time, together with a success probability. A success probability can be seen as a tail bound $\Pr(T \leq t)$ which expresses the probability that the random optimisation time T is less that a given amount of time.

Definition 2.10 (Adapted from Definition 1 in Wegener, 2005)**.** *Let A be a randomised search heuristic running for a polynomial number of rounds $p(m)$ and let $s(m)$ be the success probability, i.e., the probability that A finds an optimal search point within this phase. A is called*

- *successful, if $s(m)$ is polynomially small,*
- *highly successful, if $1 - s(m)$ is polynomially small, and*
- *successful with overwhelming probability, if $1 - s(m)$ is exponentially small.*

The reason behind calling successful the case where the success probability is just polynomially small is that, multistart variants of the algorithm which do not depend on p and s will be successful with overwhelming probability (Wegener, 2005).

Finally we introduce some methods for the time complexity analysis of EAs. Some parts of the following subsections have been freely adapted from the main three textbooks regarding the theoretical runtime analysis of EAs: Auger and Doerr (2011), Jansen (2013) and Neumann and Witt (2010).

2.3.1 Markov Chains

Markov processes are one of the most studied processes in probability theory. They are named after Andrey Markov who first studied this process in 1906. A century later, Markovian processes constitute a main component in many research fields, including computer science or physics among others (Levin et al., 2008). We briefly recap just the basic notions of Markov Chains. We refer the interested reader to Chapter 4 from the textbook by Ross (1996) or Chapter 1 from the textbook by Levin et al. (2008).

Let $\{X_t, t = 0, 1, 2, \ldots\}$ be a collection of random variables in time that takes values on a finite state space Ω. We write $X_t = x$ and say that the process is at state x at time t. We will say that such process is Markovian when the transition probability $p_{x\to y}$ of moving between any two states x and y only depends on the present state. Mathematically speaking,

$$p_{x\to y} = \Pr(X_{t+1} = y \mid X_t = x_t, X_{t-1} = x_{t-1}, \ldots X_0 = x_0) = \Pr(X_{t+1} = x \mid X_t = y). \quad (2.4)$$

Or in matrix formulation, we can assign $p_{x\to y}$ to the element (x, y) of the so-called *transition matrix P*. Analogous to transition probabilities, we can define τ-iterations transition probabilities as

$$p^{\tau}_{x\to y} = \Pr(X_{t+\tau} = x \mid X_t = y), \quad \tau \in \mathbb{N}. \quad (2.5)$$

A Markov chain is called *irreducible* if every state can be reached from every other state in finite time, i.e., $p^{\tau}_{x\to y} > 0$. Irreducible Markov chains have a *stationary distribution* $\pi \in \Omega$ such that $\pi = \pi P$ and $\pi(x) > 0$ for all $x \in \Omega$ (see e.g. Proposition 1.14 in Levin et al., 2008). A common approach to derive the stationary distribution of a Markov chain is to use the fact that π fulfils the so-called *detailed balance condition* (see e.g. Proposition 1.19 in Levin et al., 2008).

$$\pi(x) \cdot p_{x\to y} = \pi(y) \cdot p_{y\to x}, \quad \text{for all} \quad x, y \in \Omega. \quad (2.6)$$

As an example, the following theorem derives the stationary distribution of the Metropolis algorithm. This result is well known in the literature, and its somehow trivial since the algorithm was built around the Boltzmann distribution (Metropolis et al., 1953).

Theorem 2.2. *Consider the Metropolis algorithm (Algorithm 2.7) as a Markov chain on the state space $\Omega = \{0,1\}^n$ of a fitness function $f : \Omega \to \mathbb{R}$. Then, the stationary distribution of such process is*

$$\pi(x) = \frac{e^{\alpha f(x)}}{Z},$$

where $Z = \sum_{x\in\{0,1\}^n} e^{\alpha f(x)}$ is the normalisation constant.

Proof. To use the detailed balance condition, we first need to establish that the Metropolis algorithm is an irreducible Markov chain. First, we notice that local mutations can only produce (in one iteration) search points that differ in exactly one bit. However, it is straightforward to notice that after n iterations there will be a non-zero probability of mutation having produced any state of Ω. If we pessimistically assume that during this n steps mutation only produces search points with the biggest fitness difference $\Delta f^{\text{Max}} := \max_{x,y\in\Omega}(f(y) - f(x))$, we can lower bound the n-th transition probability by

$$p^n_{x\to y} \geq \left(\frac{1}{n} \cdot p^{\text{MA}}_{\text{acc}}(-\Delta f^{\text{Max}})\right)^n \geq 0,$$

where in the last inequality we have used the fact that the problem size n is finite and that $p^{\text{MA}}_{\text{acc}}(i) > 0$ for any finite $i \in \mathbb{R}$.

Now that the irreducibility of the Markov chain is proven, we can use the detailed balance condition (Equation (2.6)). Let us now consider the term $\pi(x) \cdot p_{x\to y}$, as the algorithm itself, the transition probability $p_{x\to y}$ can be broken down in two components: mutation and selection. We denote by $\text{mut}(x,y)$ the probability of mutation producing the state y from the state x. Hence, using the candidate distribution from the statement we obtain

$$\begin{aligned}\pi(x) \cdot p_{x\to y} &= \frac{e^{\alpha f(x)}}{Z} \cdot p_{x\to y} \\ &= \frac{e^{\alpha f(x)}}{Z} \cdot \text{mut}(x,y) \cdot p^{\text{MA}}_{\text{acc}}(f(y) - f(x)).\end{aligned}$$

If $f(y) - f(x) < 0$, then $p_{\text{acc}}^{\text{MA}}(f(y) - f(x)) = e^{\alpha(f(y)-f(x))}$ yielding

$$\begin{aligned}\pi(x) \cdot p_{x\to y} &= \frac{e^{\alpha f(x)}}{Z} \cdot \text{mut}(x,y) \cdot e^{\alpha(f(y)-f(x))} \\ &= \frac{e^{\alpha f(y)}}{Z} \cdot \text{mut}(x,y),\end{aligned}$$

since the mutation operator is unbiased $\text{mut}(x,y) = \text{mut}(y,x)$ and we obtain

$$\pi(x) \cdot p_{x\to y} = \frac{e^{\alpha f(y)}}{Z} \cdot \text{mut}(y,x) = \pi(y) \cdot p_{y\to x}.$$

On the other hand, if $f(y) \geq f(x)$ it follows the same calculations by swapping $x \leftrightarrow y$. $\square$

It seems natural to introduce rigorous measurements of how far the process is from this limit behaviour. In order to do that, we first measure the distance between two distributions μ and ν by using the *total variation distance*.

$$\|\mu - \nu\|_{\text{TV}} := \frac{1}{2} \sum_{x\in\Omega} | \mu(x) - \nu(x) | = \max_{A\subseteq\Omega} | \mu(A) - \nu(A) | . \tag{2.7}$$

Analogously, we can estimate the time needed for the process to reach the equilibrium with the so-called *mixing time*. This is, the first point in time when the total variation distance is smaller than a predefined threshold. We have chosen a value of $1/(2e)$ for this threshold, however there are similar but different values used in the literature.

Definition 2.11 (Mixing Time)**.** *Let $p^t(x)$ be the probability distribution of an ergodic Markov Chain after t iterations. Then, the ε-mixing time is defined as $t(\varepsilon) := \min\{t \mid \|p^t(x) - \pi\|_{\text{TV}} \leq \varepsilon\}$. And the mixing time t_{mix} is the worst case until the distance is reduced to $1/(2e)$, i.e., $t_{\text{mix}} := \max_{x\in\Omega} t\,(1/(2e))$.*

Regarding the theoretical analysis of evolutionary algorithms, He and Yao (2003) introduced an analytic framework based on Markov chains. However, the generality of this approach was at the cost of its applicability. Although they derived explicit solutions for some case studies, the authors recognised that in general it might even be impossible to solve the proposed equations.

Another method based on Markov chain theory was presented by Sudholt (2011b). Here, the author exploited the link between runtime and mixing time and applied the *coupling* technique to derive estimations of the algorithm's runtime. In Chapters 7 and 8, we will partly make use of this approach and the above introduced concepts from Markov chain theory.

2.3.2 Gambler's Ruin Problem

One special case of a Markov chain that appears often enough to be interesting by itself is the so-called *Gambler's Ruin problem*. It depicts a game where two gamblers bet their money in the following way: at each iteration player 1 wins one of player's 2 dollars with probability p_1 and player 2 wins one of player's 1 dollars with probability $p_2 = 1 - p_1$.

It is straightforward to notice that we can cast this process with a simple Markov chain. The state space will be the integers up to some maximum number $\Omega = \{0, 1, \ldots, n\}$, where n accounts for the sum of each player's money (n_1 and n_2). Since only one dollar can be won (or lost) at each time step, only transition probabilities between neighbour states (in the integer line) will have a non-zero value. Furthermore, these probabilities do not depend on the state of the process and remain fixed during the game. Finally, since the game finishes when one of the two extreme states (0 and n) is reached, these two states will not have out-going probabilities, i.e., they are *absorbing states*.

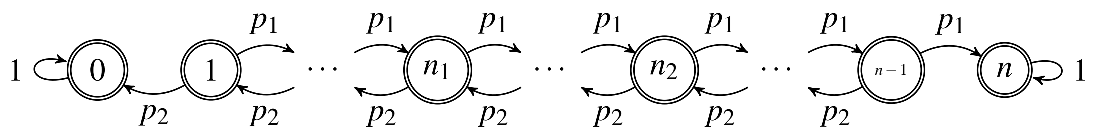


Fig. 2.7 Graphical representation of the underlying Markov Chain of a Gambler's ruin process.

The two main questions of interest for this process are: what are each player's chances of becoming bankrupt? and What is the expected time until such event occurs? The following theorem, adapted from Chapter XIV from Feller's textbook (1968), answers the above questions.

Theorem 2.3 (Gambler's Ruin). *Consider a game where two players start with $n_1 \in \mathbb{N}^+$ and $n_2 \in \mathbb{N}^+$ dollars respectively. In each iteration, player 1 wins one of player's 2 dollars with probability p_1 and player 2 wins one of player's 1 dollars with probability $p_2 = 1 - p_1$. Then the probability of player 1 winning all the dollars before going bankrupt is*

$$P_1 = \begin{cases} \frac{n_1}{n_1+n_2} & \textit{if } p_1 = p_2 \\ \frac{1-\left(\frac{p_2}{p_1}\right)^{n_1}}{1-\left(\frac{p_2}{p_1}\right)^{n_1+n_2}} & \textit{if } p_1 \neq p_2. \end{cases}$$

The expected time until either of both players become bankrupt, i.e., the expected duration of the game is

$$E[T] = \begin{cases} n_1 n_2 & \text{if } p_1 = p_2 \\ \frac{n_1-(n_1+n_2)P_1}{p_2-p_1} & \text{if } p_1 \neq p_2. \end{cases}$$

This result is useful when analysing evolutionary algorithms that are optimising a fitness path (recall Figure 2.2). This landscape feature can resemble a linear Markov chain as in a Gambler's ruin process. Finally, we present a case study to exemplify the use of this theorem for a random walk (Algorithm 2.4) on the integers.

Theorem 2.4. *Let $n/2$ be the starting position of a Random Walk that can take values on $\{0,1,\dots,n\}$. Then, the probability of reaching the state n before hitting state 0 is $1/2$. And the expected time until such event occurs will be $n^2/4$.*

Proof. Noticing that for a RW $p_1 = p_2 = 1/2$, the results follows straightforward after introducing $n_1 = n_2 = n/2$ in Theorem 2.3. □

2.3.3 The Coupon Collector Problem

Another Markov chain of interest is the so-called *coupon collector problem* (see e.g. Motwani and Raghavan, 1995). Consider a collector who wants to complete a collection of n different coupons. Whenever he makes a purchase, he chooses uniformly at random one coupon.

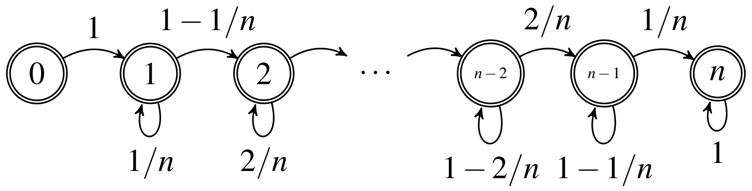


Fig. 2.8 Underlying Markov Chain of the Coupon Collector Problem.

There are two main differences to the Gambler's Ruin problem. Firstly, the collector cannot lose coupons. Secondly, the more coupons the collector has, the smaller the probability of choosing a new coupon. Obviously the collector will eventually finish the collection. Hence, the only research question is regarding the expected time until such event occurs. The following theorem, adapted from Theorem 1.21 in Auger and Doerr (2011), answers this question.

Theorem 2.5 (Coupon Collector)**.** *The expected time to collect all the coupons is $n \cdot \sum_{i=1}^{n} 1/i = nH_n$, where H_n is the nth harmonic number.*

This result is useful when analysing elitist algorithms on functions of *unitation*. This is, the class of functions where the fitness depends on the number of ones in the bitstring regardless its position. Finally, as a case study, we analyse RLS (Algorithm 2.5) on the ONEMAX problem (see Definition 2.4).

Theorem 2.6. *Consider RLS with initial search point 0^n on* ONEMAX*. Then, the expected optimisation time is $E[T] = nH_n$.*

Proof. Let x denote the number of 1-bits in the current solution. Since RLS uses local mutations, it can only create Hamming neighbours. Out of those n search points, only $n-x$ possess a higher ONEMAX fitness. The remaining neighbours will be rejected by selection if they are sampled by mutation. Hence, the probability of moving to a point of higher fitness is $p_{x\to x+1} = (n-x)/n$. Since the remaining probability falls into the current state, i.e., $p_{x\to x} = 1 - p_{x\to x+1}$, we have a coupon collector problem. Finally, calling Theorem 2.5 completes the proof. □

2.3.4 Drift Analysis

We can find the term *drift* in many scientific fields. In fluid dynamics, the *Stokes drift* is the average velocity of a fluid parcel. Also, charge carriers in a material experience a *drift velocity* due to an electric field. In population genetics, the *genetic drift* are the gene's frequency changes in a population due to a random sampling.

However, we are interested in the so-called *stochastic drift*, i.e., the expected change of a stochastic process' value. The study of the stochastic drift within the field of probability theory was started by Hajek (1982). Roughly two decades later, He and Yao (2001) introduced drift as an analysis tool for randomised algorithms.

As in classical mechanics, the velocity is the rate of change in position with respect to time, the stochastic drift is the *expected* rate of change of the stochastic process. This is a great motivation for the use of drift analysis, since classical mechanics have proven to be extraordinarily successful for analysing deterministic processes. However, the mathematical translation is not trivial since evolutionary algorithms are stochastic and discrete-time systems. Furthermore, the rules of the game vary with each algorithm and problem, unlike in nature. Nevertheless, drift is perhaps the most successful analysis tool for evolutionary algorithms.

Definition 2.12 (Stochastic Drift)**.** *Let $\{X_t\}_{t\geq 0}$ be a stochastic process over a finite state space Ω. The drift is defined as $b(X_t,t) := E[X_t - X_{t+1} \mid X_t]$.*

Although in the general case the drift can depend on the current state and the time iteration $b(X_t,t)$, in this thesis, we will only face drifts that do not depend on the current time. Hence, from now on we will write $b(X_t)$ for the sake of simplicity.

Unfortunately, there is no general method to obtain a runtime estimate from every mathematical expression of $b(X_t)$. However, there are several special cases where we can analytically solve the problem. These are, the so-called *additive* (He and Yao, 2001), *multiplicative* (Doerr et al., 2012) and *variable drift* (Johannsen, 2010) theorems. In a nutshell, the idea is to find easier expressions for the drift of the studied algorithm. These alternative drift has to be always slower (or faster) than the real drift. Then, we can derive pessimistic (or optimistic) estimations for the real runtime.

The additive drift will use constant bounds for the real drift (dashed grey lines in Figure 2.9), whereas a multiplicative drift approach consider that the drift is proportional to the current state's value (green curve in Figure 2.9). Finally, the variable drift considers a broader range of dependencies with the current state's value generalising the other approaches (red curve in Figure 2.9). However, all share an important limitation, the bounds for the drift must be monotonic functions. Hence, depending on the algorithm and problem this will yield a lack of accuracy (see gaps between the blue and other curves in Figure 2.9).

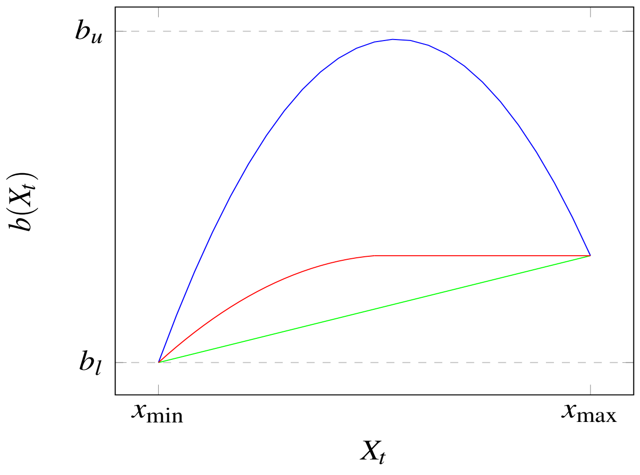


Fig. 2.9 Real drift (blue curve) of a stochastic process X_t, additive bounds b_u and b_l (dashed grey), multiplicative lower bound (green) and variable drift lower bound (red).

To exemplify the concept of drift we now compute three different lower bounds for the drift of the (1+1) EA on OneMax.

Lemma 2.2. *Let Z_t be the number of 0-bits at time t of the (1+1) EA on* OneMax. *Then, the drift $b(Z_t) = E[Z_t - Z_{t+1} \mid Z_t]$ can be bounded as follows.*

$$b(Z_t) \geq \frac{Z_t}{n}\left(1-\frac{1}{n}\right)^{n-Z_t} \geq \frac{Z_t}{en} \geq \frac{1}{en},$$

unless $Z_t = 0$, in which case, the function is already optimised.

Proof. By definition of expectation $b(Z_t) = \sum_k k \cdot \Pr(Z_t - Z_{t+1} \mid Z_t = k)$. In order to obtain a drift with an impact of k we need: mutation producing a search point with an impact of k and selection accepting the corresponding fitness change. Furthermore, we have to consider all the possible values of the impact k at state Z_t, hence:

$$b(Z_t) = \sum_{k=-(n-Z_t)}^{Z_t} k \cdot \mathrm{mut}(Z_t, Z_t - k) \cdot p_{\mathrm{acc}}^{\mathrm{EA}}(k), \tag{2.8}$$

noticing that $p_{\mathrm{acc}}^{\mathrm{EA}}$ is zero for $k < 0$ and $p_{\mathrm{acc}}^{\mathrm{EA}} = 1$ for the remaining values of k, we can simplify the previous expression to

$$b(Z_t) = \sum_{k=1}^{Z_t} k \cdot \mathrm{mut}(Z_t, Z_t - k),$$

the probability of mutation increasing the number of 1-bits by k is at least the probability of flipping k 0-bits while not flipping any 1-bit, therefore

$$\begin{aligned} b(Z_t) &\geq \sum_{k=1}^{Z_t} k \cdot \binom{Z_t}{k} \left(\frac{1}{n}\right)^k \left(1 - \frac{1}{n}\right)^{n-k} \\ &= \left(1 - \frac{1}{n}\right)^{n-Z_t} \cdot \sum_{k=1}^{Z_t} k \cdot \binom{Z_t}{k} \left(\frac{1}{n}\right)^k \left(1 - \frac{1}{n}\right)^{Z_t - k}. \end{aligned}$$

Noticing that now the sum is the expectation of a binomial distribution leads to

$$b(Z_t) \geq \frac{Z_t}{n} \cdot \left(1 - \frac{1}{n}\right)^{n-Z_t}.$$

The remaining claims follow after using $(1 - 1/n)^{n-Z_t} \geq 1/e$ since $n - Z_t < n$ and $Z_t \geq 1$. $\square$

Additive Drift

The simplest case is when the drift does not depend on the time t or the current state X_t. Here $b(X_t, t) = b$ remains constant with respect to the time and the process' value, although it can depend on the problem size n. This scenario is known as *additive drift* and it was first introduced to the EC community by He and Yao (2001).

Theorem 2.7 (Additive Drift). *Let $\{X_t\}_{t \geq 0}$ be a stochastic process over a finite set of non-negative real numbers $\Omega \subseteq \mathbb{R}_0^+$. Furthermore, let T be a random variable denoting the first point in time when $X_t = 0$. If there exist positive constants $b_u \geq b_l > 0$ such that for all t and*

X_t it holds that

$$b_u \geq E[X_t - X_{t+1} \mid X_t] \geq b_l,$$

then the expected optimisation time $E[T]$ is

$$\frac{X_0}{b_u} \leq E[T \mid X_0] \leq \frac{X_0}{b_l},$$

and by linearity of expectation

$$\frac{E[X_0]}{b_u} \leq E[T] \leq \frac{E[X_0]}{b_l}.$$

The idea here is simple, no matter how complex the drift of an algorithm is, we can find an *imaginary* process whose drift is constant and always slower (or always faster). Then, we can easily obtain a pessimistic (or optimisation) estimation of the algorithm's runtime. However, it is straightforward to notice that, depending on the real drift, this estimations can be very loose. This was informally depicted in Figure 2.9 but now we formally present an example by analysing the (1+1) EA on ONEMAX. We will see in Theorem 2.10 that the following upper bound of en^2 is not tight.

Theorem 2.8. *For any initialisation, the expected optimisation time of the (1+1) EA on* ONEMAX *is at most en^2.*

Proof. Let Z_t be the number of 0-bits at time t, then by the last bound of Lemma 2.2: $b(Z_t) \geq 1/en = b_l$. Hence, by the Additive Drift (Theorem 2.7) the runtime is at most $en \cdot \mathrm{E}[Z_0]$. Finally, pessimistically assuming that initialisation is at the all zeros bitstring we obtain $\mathrm{E}[T] \leq en^2$. □

Multiplicative Drift

As mentioned earlier, using an fixed bound for the drift can yield loose results for the runtime. In order to tighten these results, we need to relax the additive drift conditions by allowing the drift bounds to depend on the current state. This is partly solved by *multiplicative drift* by bounding the drift by an expression proportional to the current state's value. This was first presented by Doerr et al. (2012) and extended by Doerr and Goldberg (2010) to include a probability tail bound.

Theorem 2.9 (Multiplicative Drift). *Let $\{X_t\}_{t\geq 0}$ be a stochastic process over a finite set of non-negative real numbers $\Omega \subseteq \mathbb{R}_0^+$ and assume that $X_{\max} = \max\{X_t\}$ exists. Furthermore, let T be a random variable denoting the first point in time when $X_t = 0$. If there exist $\delta > 0$*

such that

$$E[X_t - X_{t+1} \mid X_t] \geq \delta X_t,$$

then the expected optimisation time $E[T]$ is

$$E[T] \leq \frac{1}{\delta}(1 + \ln X_{\max}),$$

and for every $c > 0$

$$\Pr\left(T > \frac{1}{\delta}(\ln X_{\max} + c)\right) \leq e^{-c}.$$

The condition of the drift bound being of the form δX_t might sound too specific, however, we can find this expression in many scenarios. For example, when the fitness depends only on the number of ones (unitation), a mutation operator that does not prefer flipping a 0-bit over a 1-bit (unbiased) will exhibit this effect. This family of operators tend towards sampling points with the same number of 1 and 0-bits. Hence, when the current solution X_t has $H(X_t, 0^n) \geq n/2$ the bigger the number of 1-bits, the bigger the time to obtain a new 1-bit.

Finally, we recover the same study case as in Theorem 2.8, but by using the multiplicative drift we can improve our runtime estimation from en^2 to $O(n \ln n)$.

Theorem 2.10. *For any initialisation, the expected optimisation time of the (1+1) EA on* ONEMAX *is at most $en \ln n + en$.*

Proof. Let Z_t be the number of 0-bits at time t, then by the second bound of Lemma 2.2: $b(Z_t) \geq Z_t/en$. Hence, by the Multiplicative Drift (Theorem 2.9) with $\delta = 1/en$ and $X_{\max} = n$ we obtain a runtime of $\mathrm{E}[T] \leq en(1 + \ln n)$. □

Variable Drift

In order to further improve our runtime estimates, we have to leave the multiplicative drift assumption and allow for further dependencies with X_t. The *variable drift*, almost completely, solves this issue by allowing any drift bound $b(X_t) \geq h(X_t)$, as long as $h(X_t)$ is monotone increasing. This assumption is sensible, since one expects that, in general, an EA slows its progress when it is approaching the optimum (Eiben and Smith, 2015). However, there can be drift expressions which are not monotone increasing (like the example in Figure 2.9). The variable drift was first presented by Johannsen (2010), from where we take the following statement.

Theorem 2.11 (Variable Drift)**.** *Let $\{X_t\}_{t \geq 0}$ be a stochastic process over a finite set of non-negative real numbers $\Omega \subseteq \mathbb{R}_0^+$ and assume that $X_{\min} = \{X \in \Omega \mid x > 0\}$ exists. Furthermore,* *let T be the first point in time for which $X_t = 0$. If there exists a continuous and monotone increasing function $h : \mathbb{R}_0^+ \to \mathbb{R}^+$ such that for all $t < T$ it holds that*

$$E[X_t - X_{t+1} \mid X_t] \geq h(X_t).$$

Then,

$$E[T \mid X_0] \leq \frac{X_{\min}}{h(X_{\min})} + \int_{X_{\min}}^{X_0} \frac{1}{h(x)} dx.$$

Again, we exemplify the use of the variable drift on the same problem, (1+1) EA on ONEMAX, to show that it outperforms both the multiplicative and additive drift. The following theorem is adapted from Theorem 4 in Doerr et al. (2011).

Theorem 2.12. *For any initialisation, the expected optimisation time of the (1+1) EA on* ONEMAX *is at most $en \ln n - n(e\gamma - 1) + e(e-1)$, where $\gamma = 0.577\ldots$ is the Euler-Mascheroni constant.*

Proof. Let Z_t be the number of 0-bits at time t, then by the first bound of Lemma 2.2: $b(Z_t) \geq Z_t/n(1-1/n)^{n-Z_t}$. Since this bound is monotone increasing with Z_t, let it be our h-function in the Variable Drift (Theorem 2.11). Pessimistically assuming that initialisation is at the all zeros bitstring $\mathrm{E}[T] \leq \mathrm{E}[T \mid Z_0 = n]$ and hence

$$\begin{aligned}
\mathrm{E}[T] &\leq \frac{1}{h(1)} + \int_1^n \frac{1}{h(z)} dz \\
&= \frac{n}{\left(1-\frac{1}{n}\right)^{n-1}} + \frac{n}{\left(1-\frac{1}{n}\right)^{n}} \cdot \int_1^n \frac{\left(1-\frac{1}{n}\right)^z}{z} dz \\
&\leq \frac{n}{\left(1-\frac{1}{n}\right)^{n-1}} + \frac{n}{\left(1-\frac{1}{n}\right)^{n}} \cdot \int_1^n \frac{e^{-z/n}}{z} dz. \qquad (2.9)
\end{aligned}$$

Computing a precise upper bound for the integral above is quite involved. Since this is a case study, we just recycle the following upper bound from Theorem 4 by Doerr et al. (2011).

$$\frac{n}{\left(1-\frac{1}{n}\right)^{n}} \cdot \int_1^n \frac{e^{-z/n}}{z} dz \leq n\left(1-\frac{1}{n}\right)^{-n} \left(-\gamma + \ln n + \frac{e-1}{n} - \int_1^n \frac{e}{z} dz\right).$$

Introducing this back into Equation (2.9) we obtain

$$\begin{aligned}
\mathrm{E}[T] &\leq n\left(1-\frac{1}{n}\right)^{1-n}+n\left(1-\frac{1}{n}\right)^{-n}\left(-\gamma+\ln n+\frac{e-1}{n}-\int_1^n \frac{e}{z}dz\right)\\
&\leq n+ne\left(-\gamma+\ln n+\frac{e-1}{n}\right)\\
&= en\ln n-n(e\gamma-1)+e(e-1).
\end{aligned}$$

□

Negative Drift

Finally, we present a drift theorem with a different purpose. Instead of deriving upper bounds to show the efficiency of an algorithm, the *negative drift* shows the inefficiency of algorithms by providing an exponential lower bound. There are two simple requirements for an algorithm to be inefficient: the drift must be smaller than a negative constant and the probability of large jumps must be low.

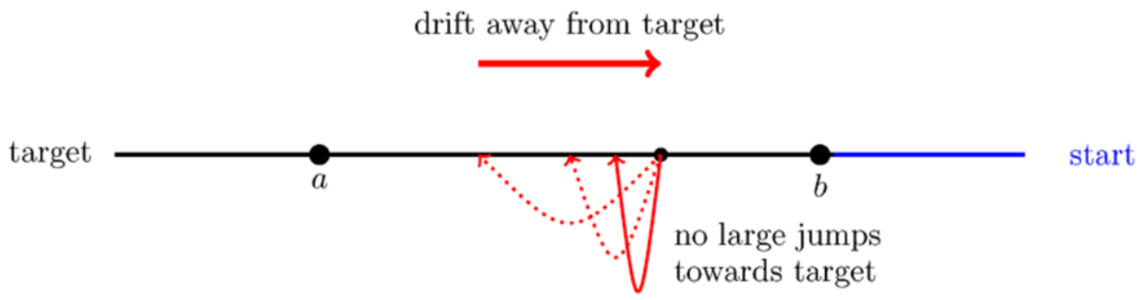


Fig. 2.10 Illustration of the scenario underlying the drift theorems for lower bounds (Figure 1 in Oliveto and Witt, 2011)

The negative drift was introduced to the EC community by Oliveto and Witt (2008); Oliveto and Witt (2012). However, we use the following formulation, adapted from Theorem 3 by Rowe and Sudholt (2014), because it considers self-loop probabilities.

Theorem 2.13 (Negative Drift with Self-loops). *Let $\{X_t\}_{t\geq 0}$ be a stochastic process over the state space $\{0,\ldots,m\}$ and transition probabilities $p_{i\to j}$. If there exist integers a,b with $0<a<b\leq m$ and $\varepsilon>0$ such that for all $a\leq k\leq b$ it holds that*

$$E[X_t - X_{t+1} \mid X_t = k] < -\varepsilon(1-p_{k\to k}),$$

and also there exist constants $r, \delta > 0$ (i. e. they are independent of m) such that for all $k \geq 1$ and all $d \geq 1$

$$p_{k \to k \pm d} \leq \frac{r(1 - p_{k \to k})}{(1+\delta)^d}.$$

Then, there is a constant $c > 0$ such that

$$\Pr\left(T \leq 2^{c(b-a)/r}\right) = 2^{-\Omega((b-a)/r)}.$$

where T be the first hitting time of a state at most a, starting from $X_0 \geq b$.

This time we consider a different heuristic, the Metropolis algorithm. The following theorem shows that if the parameter is not chosen properly, MA cannot even optimise a simple function like ONEMAX.

Theorem 2.14. *If $\alpha \leq (1-\varepsilon)\ln n$ for some $0 < \varepsilon < 1$, then the expected optimisation time of the Metropolis algorithm on* ONEMAX *is at least 2^{cn} with overwhelming probability $1 - 2^{-\Omega(n)}$, for some constant $c > 0$ and a initial number of ones $X_0 \leq 3n/4$.*

Proof. Let X_t be the number of 1-bits at time t. To compute the drift, we follow similar steps as in the proof of Lemma 2.2. We start from adapting Equation (2.8) to the Metropolis algorithm, yielding

$$b(X_t) = \sum_{k=-X_t}^{n-X_t} k \cdot \mathrm{mut}\,(X_t, X_t + k) \cdot p_{\mathrm{acc}}^{\mathrm{MA}}(k),$$

since MA uses local mutations, it can only create points with Hamming distance 1. Hence $k = \{-1, 1\}$, which yields

$$b(X_t) = \mathrm{mut}\,(X_t, X_t + 1) \cdot p_{\mathrm{acc}}^{\mathrm{MA}}(1) - \mathrm{mut}\,(X_t, X_t - 1) \cdot p_{\mathrm{acc}}^{\mathrm{MA}}(-1),$$

out of the n Hamming neighbours only X_t have a worse fitness, hence the mutation probabilities become

$$b(X_t) = \frac{n - X_t}{n} \cdot p_{\mathrm{acc}}^{\mathrm{MA}}(1) - \frac{X_t}{n} \cdot p_{\mathrm{acc}}^{\mathrm{MA}}(-1)$$

using MA's acceptance probability (Equation (2.3)) leads to

$$b(X_t) = \frac{n - X_t}{n} - \frac{X_t}{n} \cdot e^{-\alpha} = 1 - \frac{X_t}{n}\left(1 + e^{-\alpha}\right).$$

Finally, we introduce the statement's condition $\alpha \leq (1-\varepsilon)\ln n$ yielding

$$b(X_t) \leq 1 - \frac{X_t}{n}\left(1 + \frac{1}{1-\varepsilon}\right).$$

Let us now consider when the algorithm is optimising the last $n/4$ 1-bits. Then, $b(X_t \geq 3n/4) \leq 1 - 3/4(1+1/(1-\varepsilon))$, which is negative for any $0 < \varepsilon < 1$ fulfilling the first condition from the Negative Drift with Self-loops (Theorem 2.13). The second condition holds trivially since the probability of jumps further than the Hamming neighbourhood have zero probability. □

2.4 Fixed Budget

In addition to runtime analysis, Jansen and Zarges (2012) introduced a new approach to measure the performance of randomised algorithms, the *fixed-budget* perspective. Runtime analysis, as shown in the previous section, focuses on the expected first hitting time of the optimum, this time is expressed as a function of the problem size; $\mathrm{E}[T] = f(n)$. On the other hand, fixed-budget considers the expected fitness X_t obtained after t iterations. Hence, the result will also depend on the time; $\mathrm{E}[X_t] = f(n,t)$. As argued by the authors, this approach provides more information about the optimisation process than the classical runtime analysis. Furthermore, it is closer to experimental settings where one typically runs the algorithms for a fixed mount of time (budget).

Nevertheless, both approaches should be in concordance. If we know the first hitting time $T(X)$ of any specific fitness value X, it looks natural to compute the inverse function to obtain the equivalent fixed-budget result. As shown by Doerr et al. (2013) this is the case for deterministic algorithms, but it will not be necessarily true for randomised algorithms. The authors circumvent this problem by using sharp bounds on the probability of deviating from the expected times. Finally, the authors developed a method to recycle expected optimisation times to derive fixed-budget results.

Fixed budget has kept gaining importance in the community with roughly a publication per year. Nallaperuma et al. (2017a) produced the first fixed-budget results for a real-world NP-hard problem, the *travelling salesman problem*. Lengler and Spooner (2015) derived the fixed-budget analogue of the multiplicative drift (Theorem 2.9). Doerr et al. (2016) studied optimal parameter choices for both runtime and fixed-budget. And Paixão and Pérez Heredia (2017) derived additive and multiplicative drift theorems based on stochastic differential equations (see Section 8.3).

Finally, we conclude this section with an example application. Since we already have drift estimates for the (1+1) EA on ONEMAX and for the sake of brevity, we recover the same case study as in the previous section.

Theorem 2.15. *Let $\{X_t\}_{t\geq 0}$ be the number of 1-bits in the current solution of the (1+1) EA on* ONEMAX. *Then, after t iterations*

$$E[X_t] \geq n\left(1 - e^{-\frac{t}{en}}\right) + E[X_0] \cdot e^{-\frac{t}{en}}.$$

In order to prove the previous claim, we use the following Fixed-Budget Multiplicative Drift Theorem. The statement corresponds to Theorem 1 by Lengler and Spooner (2015).

Theorem 2.16. *Let $\{X_t\}_{t\geq 0}$ be a stochastic process over a finite set of non-negative real numbers $\Omega \subseteq \mathbb{R}_0^+$, and $0 < b_l < b_u < 1$. If for all t it holds that,*

$$b_u X_t \geq E[X_t - X_{t+1} \mid X_t] \geq b_l X_t.$$

Then (for $b_u \leq 0.797$) it also holds that

$$X_0 e^{-2b_u t} \leq X_0(1-b_u)^t \leq E[X_t \mid X_0] \leq X_0(1-b_l)^t \leq X_0 e^{-b_l t}.$$

Proof of Theorem 2.15. Let Z_t be the number of 0-bits at time t, then by the second bound of Lemma 2.2: $b(Z_t) \geq Z_t/en$. Hence, by the Fixed-Budget Multiplicative Drift (Theorem 2.16) with $b_l = 1/en$ we obtain

$$\mathrm{E}[Z_t \mid Z_0] \leq Z_0 e^{-t/en}.$$

Translating to the number of ones ($X_t = n - Z_t$) leads to

$$\begin{aligned} n - \mathrm{E}[X_t \mid X_0] &\leq (n - X_0)e^{-t/en} \\ \mathrm{E}[X_t \mid X_0] &\geq n - (n - X_0)e^{-t/en} \\ \mathrm{E}[X_t \mid X_0] &\geq n(1 - e^{-t/en}) + X_0 \cdot e^{-t/en}. \end{aligned}$$

Applying linearity of expectations completes the proof. □

Chapter 3

Population Genetics

To finalise the introductory part of this thesis, we present a chapter about *Population Genetics*. The main scope in population genetics is to study the variations in gene frequencies due to evolutionary processes such as mutation, recombination or selection. This chapter does not aim to be a thorough literature review or a full introduction to the field. The scope of this chapter is to motivate, from the biological point of view, the results that we will present on Parts II and III. Hence, we will restrict this chapter merely to the mathematical and biological foundations needed to derive the so-called *Strong Selection Weak Mutation* (SSWM) regime. The choice of this regime is not arbitrary, during this chapter the importance and relevance of the SSWM regime for the theoretical study of evolution will become clear. For a general introduction to the concepts of population genetics we refer the interested reader to the textbook by Barton et al. (2007). In addition, for a proper introduction to the mathematics and methods of PG we recommend the textbook by Rice (2004). To finalise this disclaimer, we would like to point out that we have chosen to use the same notation as in the previous chapter, therefore, it can substantially differ from the notation used in the PG literature.

We have followed the standard approach of many PG texts, however there are deviations and simplifications for the above mentioned reasons. After briefly going through the history of Population Genetics, we will focus on the independent study of selection, mutation and genetic drift (Section 3.1). Along the way, our analysis will highlight the importance of keeping track of the frequency of non-optimal alleles in the population. In Section 3.2, we generalise the analysis to combine selection, mutation and genetic drift altogether in one model. This approach will yield an important result: detrimental alleles not only can avoid extinction, but that they can even take over the population by replacing all the genotypes. Finally, we conclude with the derivation of the SSWM regime which will become a crucial component of this thesis.

Before jumping into the mathematical foundations of PG, we briefly recap the history of this field. As mentioned in the introduction, we can trace back the human interest for evolution to the ancient Greeks (e.g. Hippocrates, circa 400 BCE or Aristotle, circa 300 BCE), specifically to the concept of *inheritance of acquired characteristics*. This concept claims that an individual can transfer the physiological changes acquired during its lifetime to its offspring. Many centuries later, Lamarck (1809) recovered this idea in, what can be considered, the start of the scientific study of evolution.

Fifty years later, Darwin (1859) and Wallace (1855) independently proved wrong Lamarck's proposed evolutionary mechanism. After reading the work on population growth of the economist Malthus (1798), Darwin realised that while natural resources are limited, species are reproducing exponentially. Hence, individuals are in competition for resources and if the variations present in living organisms are hereditary, *nature will be selecting the fitter individuals*.

However, Darwin could not solve what appeared to be a major flow in his theory. If parents transfer a blend of their characteristics to their children, diversity should vanish after sufficiently many generations. Interestingly enough, the answer was hidden in the already published work by Mendel (1866). Nevertheless, we had to wait until the early 1900s, when Hardy (1908) and Weinberg (1908) independently showed with a simple formula that Mendelian inheritance does preserve diversity under random mating. Later in the 1920s and 1930s, Fisher (1930), Haldane (1990) and Wright (1932) developed a precise mathematical framework that would definitely unify Mendelian genetics and Darwian evolution, giving birth to the research field of population genetics.

Finally, we would like to point out that despite being a theoretical discipline, the applications and repercussions of PG in the real world are huge. Some examples collected by Nowak (2006) are: HIV progression mechanism, evolution of virulence, cancer evolutionary dynamics or language evolution.

3.1 Selection, Mutation and Genetic Drift

In this section we introduce the basic notions of selection, mutation and genetic drift within the framework established by Fisher (1930), Haldane (1990) and Wright (1932). This approach studies evolution by assigning fitness values to genotypes and analysing the change in allele frequencies. Although this is not the only approach, it has proven to be very useful and has greatly contributed to the basic understanding of evolution (Rice, 2004). It is important to note that the term *genetic drift* differs from the stochastic drift introduced in

Definition 2.12. Although in biology, genetic drift is typically referred to just as *drift*, we will use the full name to avoid confusion.

3.1.1 Selection

As mentioned above, our aim is to analyse the change in allele frequencies. However, selection acts mainly at the genotype level[1] and therefore we have to establish a relationship between allele and genotype frequencies. Hardy (1908) and Weinberg (1908) independently showed that under random mating (and absence of mutation, selection and migration) the genotype frequencies only depend on the allele frequencies. Furthermore, they remain constant during following generations, proving that diversity is preserved (under those conditions).

Definition 3.1 (Hardy-Weinberg (H-W) Frequencies)**.** *Let A_i be an allele i and p_i its frequency. Then its genotype frequencies* $\mathrm{Freq}(A_iA_j)$ *are*

$$\begin{aligned}
\mathrm{Freq}(A_iA_i) &= p_i^2 \\
\mathrm{Freq}(A_iA_j) &= 2p_ip_j \quad (j \neq i) \\
\mathrm{Freq}(A_jA_j) &= p_j^2.
\end{aligned}$$

Note that the order of the alleles in a genotype does not matter since it only represents if the allele comes from the mother or from the father. Then we do not distinguish between heterozygotes $(A_iA_j = A_jA_i)$ and hence the factor 2 in the frequency $\mathrm{Freq}(A_iA_j)$. If we were interested in knowing the origin of the allele A_i, we would have to distinguish between A_iA_j and A_jA_i, and also between their frequencies $\mathrm{Freq}(A_iA_j)$ and $\mathrm{Freq}(A_jA_i)$. However, in most populations the number of genotypes A_jA_i is the same as the number of genotypes A_iA_j and the distinction is not necessary (see e.g. Chapter 2 in Crow and Kimura, 2009).

With the relation between genotype and allele frequencies we just have to define the notion of fitness within biology. The following definition is adapted from Chapter 2 in the textbook by Rice (2004).

Definition 3.2 (Fitness)**.** *Let P be a population of individuals with genotypes A_iA_j. Then we can define the following fitness measurements:*

- ***Fitness of an individual,** F: is the reproductive contribution of an individual to the next generation.*

[1]Strictly speaking it acts on individuals through their phenotype, which is determined by the genotype.

- ***Genotypic fitness,*** f_{ij}***:*** *is the average fitness of all individuals with the genotype* A_{ij} *in a population.*

- ***Marginal fitness,*** f_i^**: is the average genotypic fitness over all the genotypes that contains the allele* A_i*, weighted by the probability of finding such allele in that genotype.*

Let us now focus on the most common selection model studied in PG: selection acting on one locus with two alleles (Rice, 2004). Hence, we can simplify the notation from Definition 3.1 to: just p for the frequency of the allele A_1 and obviously the other allele will have a frequency of $1-p$.

We have defined the mean fitness of a population in terms of how much it favours the growth of such population, it seems natural then, to measure this growth per generation. Let N_t be the size of a population at time t, we can compute the population size after one generation as the sum of each genotype's fitness, weighted by their frequency in the population. Assuming H-W frequencies, this is simply $N_{t+1} = N_t\overline{f}$, where $\overline{f}$ is the *mean population fitness*.

$$\overline{f} = p^2 f_{11} + 2p(1-p)f_{12} + (1-p)^2 f_{22}. \tag{3.1}$$

Recall that since the order of the alleles in the genotype is not relevant we have that $f_{21} = f_{12}$. We now take the derivative of the previous equation to study how $\overline{f}$ varies with respect to the allele frequencies. For that, we have to consider the case where the fitness is a function of the allele frequencies, hence

$$\frac{d\overline{f}}{dp} = 2pf_{11} + p^2\frac{df_{11}}{dp} + 2(1-2p)f_{12} + 2p(1-p)\frac{df_{12}}{dp} - 2(1-p)f_{22} + (1-p)^2\frac{df_{22}}{dp}. \tag{3.2}$$

To simplify this expression we recover the concept of *marginal fitness* (see Definition 3.2), f_i^* can be computed as the probability of finding A_i in each genotype. If we stick to the assumption of random mating used to derived the H-W frequencies, these probabilities are simply each allele frequency. Therefore,

$$\begin{aligned} f_1^* &= pf_{11} + (1-p)f_{12} \\ f_2^* &= pf_{12} + (1-p)f_{22}. \end{aligned} \tag{3.3}$$

Introducing this back into Equation (3.2) leads to

$$\frac{d\overline{f}}{dp} = 2(f_1^* - f_2^*) + p^2\frac{df_{11}}{dp} + 2p(1-p)\frac{df_{12}}{dp} + (1-p)^2\frac{df_{22}}{dp},$$

which we can simplify by noticing that the three last terms are just $\mathrm{E}\left[df/dp\right]$ since each derivative is multiplied by its corresponding H-W frequency, leading to

$$\frac{d\overline{f}}{dp} = 2\left(f_1^* - f_2^*\right) + \mathrm{E}\left[\frac{df}{dp}\right]. \tag{3.4}$$

Let us leave this equation for a moment and focus again on the marginal fitnesses, from Definition 3.2, we can also understand f_i^* as the expected number of descendants of the allele A_i. Hence, if at the current generation there are n_i alleles of type A_i, in the next generation there will be, in expectation, $n_1 f_1^*$. Furthermore, we also notice that the total population size $N = \sum n_i$ grows with the mean population fitness $\overline{f}$. Then, we can derive the allele frequencies in the next generation as

$$p_{t+1} = n_1 f_1^* / N\overline{f} = p_t f_1^* / \overline{f}.$$

Returning to our original aim, we compute the variation in the allele frequencies $\Delta p = p_{t+1} - p_t$ as follows

$$\Delta p = \frac{p\left(f_1^* - \overline{f}\right)}{\overline{f}} = \frac{p(1-p)(f_1^* - f_2^*)}{\overline{f}}, \tag{3.5}$$

where in the last step we have used the fact that $\overline{f} = pf_1^* + (1-p)f_2^*$. Finally, we recover Equation (3.4) to introduce it into Equation 3.5 yielding

$$\Delta p = \frac{p(1-p)}{2\overline{f}}\left(\frac{d\overline{f}}{dp} - \mathrm{E}\left[\frac{df}{dp}\right]\right). \tag{3.6}$$

This equation is crucial for the theoretical understanding of evolution. In the special case when the fitness is frequency-independent (i.e, $df/dp = 0$) it corresponds with the equation for an *adaptive landscape* by Wright (1937).

$$\Delta p = \frac{p(1-p)}{2\overline{f}}\frac{d\overline{f}}{dp}. \tag{3.7}$$

If we consider this equation as a dynamical system, we find a very important result. Apart from the extreme cases when one of the two alleles has taken over the whole population ($p = 0$ or $p = 1$), the fixed points of Equation (3.7) (i.e., when $\Delta p = 0$) are located at $d\overline{f}/dp = 0$. Therefore, we find stability when the mean population fitness is maximised! This result backs up the principle of *adaptation* from the Darwin-Wallace evolutionary theory. Unfortunately, Moran (1963) showed that for the multi-loci case it no longer holds (in general) that populations tend to maximize $\overline{f}$. Specifically, when the fitness is dependent on more than one locus. However, Moran also mentioned that adaptive landscapes can still be used to obtain approximated results when the genotypic fitnesses are very close to each other.

To further understand Equation (3.7), we present a case study where the heterozygote has a higher fitness than both homozygotes. This is modelled by assigning (for example) the following genotypic fitnesses: $f_{11} = f_{22} = 1$ and $f_{12} = 1 + \Delta f$. Then, assuming H-W frequencies, we can use Equation (3.1) to compute the mean population fitness: $\overline{f} = 1 + 2\Delta f p(1-p)$. Introducing this value of $\overline{f}$ in Equation (3.7) leads to

$$\Delta p = \frac{p(1-p)(1-2p)\Delta f}{1+2\Delta f p(1-p)}. \tag{3.8}$$

Let us now plot both Δp and $\overline{f}$ against p. In the right-hand graph of Figure 3.1, we can observe that the mean population fitness approaches its minimum value when there are no alleles A_1 ($p = 0$). This corresponds with all the population consisting of copies of the genotype A_{22}. Analogously, when $p = 1$, all the alleles present in the population are A_{11}, hence a minimal $\overline{f} = 1$. However, when the population is composed of half A_1 alleles ($p = 1/2$) we observe that the fitness is maximised.

As we mentioned before, the value $p = 1/2$ must correspond with a stability point ($\Delta p = 0$). We can observe that this is the case on the left-graph of Figure 3.1. It is also interesting to study the stability of this fixed point. For that, we observe that $\overline{f}(p)$ is increasing for $p < 1/2$ and decreasing as soon as $p > 1/2$. Since the sign of Δp is determined by the slope of $\overline{f}(p)$, recall Equation (3.7), we can conclude that the point $p = 1/2$ is a stable equilibrium point.

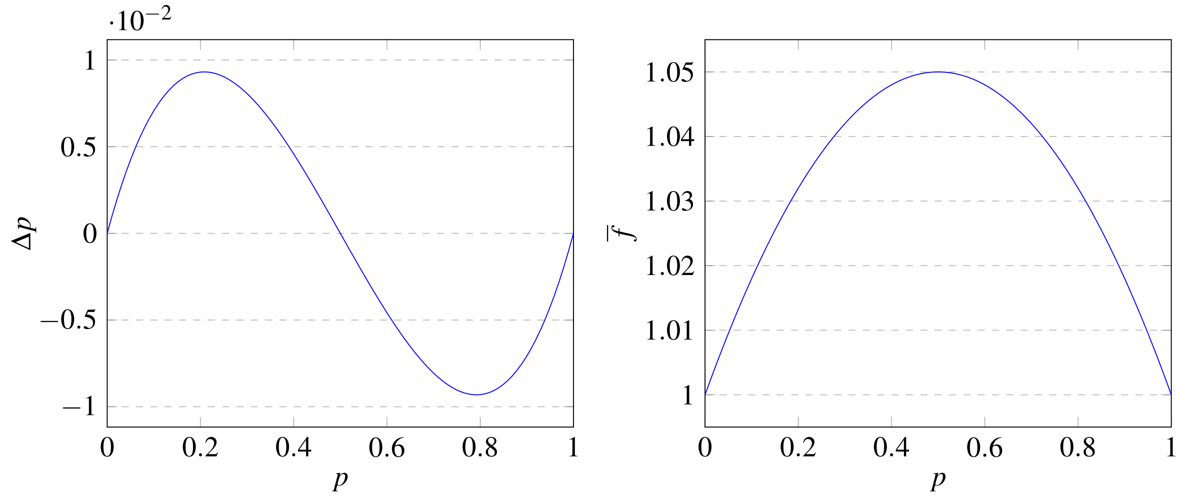


Fig. 3.1 Heterozygote advantage model with $f_{11} = f_{22} = 1$ and $f_{12} = 1.1$. The graph on the left shows the variation in the frequency $\Delta p = p_{t+1} - p$ of allele A_1, versus the frequency p of the allele A_1. The graph on the right shows the mean population fitness $\overline{f}$ against p.

Finally, it is important to notice that if we recover Equation (3.6), then the fitness has a dependency with the allele frequencies and we see that it is not necessarily the case that $\overline{f}$ is maximised. This presence of non-optimal genotypes prevents the population from reaching the maximum fitness. To gain further insight into this issue we need to consider the mechanism that produces new genotypes, *mutation*.

3.1.2 Mutation

Mutation is the main mechanism to produce diversity. Diversity already appeared in the previous subsection, we saw how different alleles may have a different genotypic fitness, but we did not wonder about the origin of such difference. Equation (3.6) yields an important insight, despite having a selecting disadvantage (lower fitness), a non-optimal genotype can survive in the population. We now wish to study this trade-off between mutation producing a deleterious allele and selection “trying to get rid of it”. More precisely, we would like to derive an expression for when these two factors cancel out, this is known as the *mutation-selection equilibrium*. This is an important research question since it is one of the most important factors maintaining diversity within a population (Rice, 2004).

Since we want to study the variation in allele frequencies, we recover Equation (3.5). To study mutation, we introduce terms for the rate at which each allele mutates to the other. Considering as well the frequency of each allele leads to

$$\Delta p = p\frac{f_1^* - \overline{f}}{\overline{f}} + \mathrm{mut}\left(A_2,A_1\right)(1-p) - \mathrm{mut}\left(A_1,A_2\right)p, \tag{3.9}$$

where $\mathrm{mut}\left(A_i,A_j\right)$ denotes the rate at which the allele A_i mutates into the allele A_j. To further continue studying the frequency of a detrimental allele, we have to make some assumptions on the fitness of each genotype. To model this selective difference we introduce a parameter $\Delta f = f_{22} - f_{11} > 0$ to represent the selection disadvantage of the genotype A_1A_1 against the optimal genotype. Furthermore, to include a possible dominance relationship between the two alleles, we use the parameter $h \in [0,1]$. This constitutes the traditional model to study the mutation-selection equilibrium (see Chapter 1 of Rice, 2004).

$$f_{11} = 1 - \Delta f, \quad f_{12} = 1 - h\Delta f, \quad f_{22} = 1. \tag{3.10}$$

Assuming H-W frequencies we can now compute the mean population and the marginal fitnesses as follows.

$$\begin{aligned}\overline{f} &= p^2(1-\Delta f)+2p(1-p)(1-h\Delta f)+(1-p)^2\\ &= 1-\Delta f p^2-2h\Delta f p(1-p)\\ f_1^* &= 1-\Delta f p-h\Delta f(1-p).\end{aligned} \tag{3.11}$$

The gap between $\overline{f}$ and the fitness of an optimal population (all genotypes at A_2A_2, i.e., $p=0$) is know as the *genetic load*, which in our model has a value of $\Delta f p^2+2h\Delta f p(1-p)$.

Let us consider again Equation (3.9). We could introduce the fitnesses computed in Equation (3.11). However to obtain a simple mathematical result we will perform some approximations. First, since one would expect a deleterious allele to have a really low frequency, it is a good approximation to use $\overline{f}\approx 1$. Then we obtain,

$$\begin{aligned}\Delta p &\approx p(f_1^*-1)+\mathrm{mut}\,(A_2,A_1)\,(1-p)-\mathrm{mut}\,(A_1,A_2)\,p\\ &= p(-\Delta f p-h\Delta f(1-p))+\mathrm{mut}\,(A_2,A_1)\,(1-p)-\mathrm{mut}\,(A_1,A_2)\,p.\end{aligned} \tag{3.12}$$

To find the equilibrium point p_{eq} we solve this equation for $\Delta p=0$, leading to

$$\Delta f p_{\mathrm{eq}}^2+h\Delta f p_{\mathrm{eq}}(1-p_{\mathrm{eq}})\approx \mathrm{mut}\,(A_2,A_1)\,(1-p_{\mathrm{eq}})-\mathrm{mut}\,(A_1,A_2)\,p_{\mathrm{eq}}. \tag{3.13}$$

The second approximation is to neglect square terms in p_{eq}, yielding

$$h\Delta f p_{\mathrm{eq}}\approx \mathrm{mut}\,(A_2,A_1)\,(1-p_{\mathrm{eq}})-\mathrm{mut}\,(A_1,A_2)\,p_{\mathrm{eq}}.$$

Finally, it is also safe to assume that the mutation rates are small, hence we can neglect their products with p_{eq}. Hence,

$$p_{\mathrm{eq}}\approx\frac{\mathrm{mut}\,(A_2,A_1)}{h\Delta f}. \tag{3.14}$$

We conclude with a graphical representation of this model. Unlike in the heterozygote advantage model from Figure 3.1, we can observe in Figure 3.2 (right) that the mean population fitness always decreases with p. According to Equation 3.7, this implies that generations will yield a reduction in the proportion p of deleterious alleles A_1 (Figure 3.2, left). However, for $p<p_{\mathrm{eq}}$ we observe that $\Delta p>0$ which corresponds to the situation when the mutation rate at which the allele A_1 appears is higher than the selective pressure.

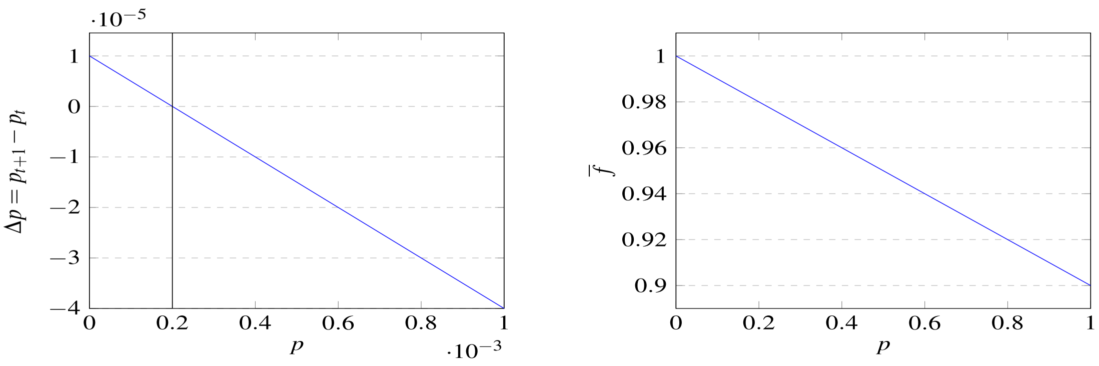


Fig. 3.2 Detrimental allele model from Equation (3.10) with $\text{mut}(A_2,A_1) = 10^{-5}$, $\Delta f = 0.1$ and $h = 1/2$. Notice that for the left plot only values for $p \in [0, 0.001]$ are considered, hence Equation (3.12) becomes $\Delta p \approx \text{mut}(A_2,A_1) - \Delta f h p$. And by Equation (3.14): $p_{\text{eq}} \approx 0.0002$.

3.1.3 Genetic Drift

So far we have assumed that the time-variation of the allele frequencies is determined by the expected change due to different evolutionary mechanisms. This would be true for infinite populations, however populations are finite and hence it is important to consider deviations from the expected behaviour. In this subsection, we will momentarily forget about selection and mutation to study just these deviations from expectation, i.e., the *genetic drift*.

The most common model to analyse the genetic drift is the so-called *Wright-Fisher model* (Ewens, 2004). This model considers that the genetic drift is due to the following random sampling of gametes (reproduction). Let p be the proportion of the allele A_1 in a population of size N ($2N$ locus). At each generation we form a new population by taking a sample (with replacement) of $2N$ alleles from the current population. Hence, the probability of the new population having i alleles A_1 is given by

$$\Pr(|A_1| = i) = \binom{2N}{i} p^i (1-p)^{2N-i}, \tag{3.15}$$

where $|A_1|$ denotes the number of A_1 alleles. This is known as *binomial sampling* since the previous expression is a binomial distribution with parameters p and $2N$. Since we are interested in deviations from the expected behaviour it is natural to look into the variance of this process, which has a value of $\text{Var}[|A_1|] = 2Np(1-p)$. But we are interested in the allele frequencies ($p = |A_1|/2N$), therefore recalling that $\text{Var}[aX] = a^2 \text{Var}[X]$ for any variable X and a constant a, we obtain

$$\text{Var}[p] = \frac{p(1-p)}{2N}. \tag{3.16}$$

This equation is important since it shows that, for a random sampling of gametes, the bigger the population size is, the smaller the deviation from the expected behaviour. Hence, while the study for large populations will be mainly described by our analysis from Subsections 3.1.1 and 3.1.2, the study of small populations will be mainly described by the genetic drift.

3.2 Diffusion Theory

In previous subsections we saw that large enough populations progress with the expected change due to mutation and selection. On the other hand, small populations are governed by the genetic drift. We wonder now what will happen for intermediate population sizes, and if there is a way to combine all the previous evolutionary processes within the same model. We find the answer in Rice (2004): *"The most powerful method devised for combining different deterministic and stochastic mechanisms in the same model is diffusion theory"*.

Diffusion theory was introduced to population genetics by Wright (1945), and it has since become one of the main methods in the field. For example, it was the method used by Kimura (1983) to derive his *neutral theory of molecular evolution*. As in evolutionary algorithms, due to the stochastic component of the evolutionary processes, we cannot explain with absolute certainty what will happen in future generations. Hence, once again, we have to rely on probability theory.

We will denote by $\phi(p,t)$ the probability that the allele A_1 is at frequency p at time t. If one considers a large number of populations (with identical conditions), $\phi(p,t)$ can also be interpreted as the distribution of allele frequencies among a large number of those populations.

Under the diffusion theory we can group the evolutionary processes according to their effect in the change of the allele frequencies of one population (Rice, 2004):

- *Directional processes:* those processes with a non-zero expectation. Includes selection, mutation, migration and recombination.

- *Non-directional processes:* those processes with no expectation. Includes random variation in mating or survival selection, i.e., all genetic drift-like processes.

The action of these processes will shape how $\phi(p,t)$ varies over time. While directional processes will cause $\phi(p,t)$ to shift along the state space p, non-directional processes will contribute to a spread out of the probability distribution.

Let $\psi(\delta \mid p)$ be the transition probability of reaching p from $p-\delta$ in the time between one generation Δt. To study the time evolution of the probability density $\phi(p,t)$ of the allele frequencies we use the so-called Chapman-Kolmogorov Equation (see e.g. Feller, 1949).

$$\phi(p,t+\Delta t) = \int \psi(\delta \mid p) \cdot \phi(p-\delta,t) \cdot d\delta. \tag{3.17}$$

If we can solve this equation, we will have a complete understanding of the process no matter the current allele frequencies or time. However, the Kolmogorov forward equation is typically intractable. In order to work with this equation we will perform the following approximations.

Firstly, despite the fact that the allele frequencies p can only take values that are multiples of $1/2N$, we will treat them as a continuous variable (the bigger the population size is, the better this approximation will be). Furthermore, instead of studying the change in p per generation we will approximate the process as a time-continuous one. This is known as the *continuous approximation* since it corresponds to a first order Taylor expansion of the term $\phi(p,t+\Delta t)$ around $\Delta t = 0$. From a direct application of Taylor's theorem, we can see that this approximation carries an error of order $O(\Delta t)$.

Secondly, we assume that the transition probability $\psi(\delta \mid p)$ is small enough such that it can be accurately approximated by a Gaussian distribution with the same expectation and variance as the original $\psi(\delta \mid p)$. This is known as a *Gaussian approximation* since it corresponds with a second order Taylor expansion of the term $\psi(\delta \mid p) \cdot \phi(p-\delta,t)$ around $p-\delta = p$. By Taylor's theorem, it will carry an error of order $O(\delta^2)$.

Obviously, these are many approximations to constitute a rigorous model (Gillespie, 1989). However, the conditions are biologically reasonable and the model proves to be good enough for many purposes. Furthermore, Karlin and McGregor (1964) showed that the model tends to be exact when N tends to infinity, provided that we scale time in units of $2N$ and $\psi(\delta \mid p)$ leads to changes in p of at most $1/(2N)$.

After performing the above approximations, one obtains the so-called *diffusion* or Kolmogorov forward equation. For simplicity and since this derivation is widely known (see e.g. Einstein, 1905) we omit them from the main text, however they can be found in Appendix B.

$$\Delta t \cdot \frac{\partial \phi(p,t)}{\partial t} \approx -\frac{\partial}{\partial p}\left(\phi(p,t) \cdot \mathrm{E}[\psi]\right) + \frac{1}{2} \cdot \frac{\partial}{\partial p^2}\left(\phi(p,t) \cdot \mathrm{E}\left[\psi^2\right]\right). \tag{3.18}$$

However, the diffusion equation still seems quite intractable. We will follow the same approach as Wright (1945) and study just the equilibrium distribution. This limit case is obtained from the previous equation when $\partial\phi(p,t)/\partial t = 0$. Hence,

$$\frac{\partial}{\partial p}\phi_{\mathrm{eq}}(p,t)\cdot \mathrm{E}[\psi] \approx \frac{1}{2}\cdot\frac{\partial}{\partial p^2}\phi_{\mathrm{eq}}(p,t)\cdot \mathrm{E}\left[\psi^2\right]. \tag{3.19}$$

Again, we include the calculations in Appendix B, but it can be shown that the solution, up to an integration constant c, is

$$\phi_{\mathrm{eq}} \approx \frac{c}{\mathrm{Var}[\psi]}e^{\int \frac{2\mathrm{E}[\psi]}{\mathrm{Var}[\psi]}dp}. \tag{3.20}$$

However, for some scenarios (we will see an example in the following subsections) it is more relevant to consider the *Kolmogorov backward equation*. To derive this equation, we must rewrite Equation (3.17) (Chapman-Kolmogorov equation), by taking into consideration the starting state of the allele frequencies $p(t=0) := p_0$. Hence, $\phi(p,t \mid p_0)$ now reads as the probability density of populations with allele frequency p at time t given that the frequency at time 0 was p_0.

$$\phi(p,t+\Delta t \mid p_0) = \int \psi(\delta \mid p_0)\cdot\phi(p,t \mid p_0+\delta)\cdot d\delta. \tag{3.21}$$

Following similar steps as for the forward equation, we obtain analogous expressions to: the diffusion equation, the diffusion equation at equilibrium and its solution. Once more, all the calculations are included in Appendix B.

$$\Delta t\cdot\frac{\partial\phi(p,t \mid p_0)}{\partial t} \approx \mathrm{E}[\psi]\cdot\frac{\partial\phi(p,t \mid p_0)}{\partial p_0} + \frac{\mathrm{Var}[\psi]}{2}\cdot\frac{\partial\phi(p,t \mid p_0)}{\partial p_0^2} \tag{3.22}$$

$$\mathrm{E}[\psi]\cdot\frac{\partial\phi_{\mathrm{eq}}(p,t \mid p_0)}{\partial p_0} + \frac{\mathrm{Var}[\psi]}{2}\cdot\frac{\partial\phi_{\mathrm{eq}}(p,t \mid p_0)}{\partial p_0^2} \approx 0 \tag{3.23}$$

$$\frac{\partial\phi}{\partial p_0} \approx c\cdot e^{-\int \frac{2\mathrm{E}[\psi]}{\mathrm{Var}[\psi]}dp}, \tag{3.24}$$

where c is an integration constant. It is straightforward to notice that we have not completely solved Equation (3.23), we still need to perform one more integration step. Due to the already involved expression of Equation (3.24), we leave it here and we will solve it for specific scenarios in subsequent subsections.

3.2.1 Mutation, Selection and Genetic Drift under Diffusion Theory

We are now ready to include in the same model: mutation, selection and the genetic drift. In order to do that, we recover the results from the previous subsections where we analysed separately these processes.

Let us start considering the equilibrium distribution obtained in Equation (3.20). As mentioned earlier, directional processes (mutation and selection) are those that yield a non-zero expected change, hence they will be our estimates for $\mathrm{E}[\psi]$. By using Equation (3.7) (Wright's adaptive landscape) and including the mutation rates as we did in Equation (3.9), we obtain

$$\mathrm{E}[\psi] = \frac{p(1-p)}{2\overline{f}}\frac{d\overline{f}}{dp} + \mathrm{mut}(A_2,A_1)(1-p) - \mathrm{mut}(A_1,A_2)\,p. \tag{3.25}$$

Analogously, non-directional processes (genetic drift) will be the causing the spread out of the allele frequencies, i.e., they will contribute to $\mathrm{Var}[\psi]$. By using Equation (3.16) (Wright-Fisher model) we obtain

$$\mathrm{Var}[\psi] = \frac{p(1-p)}{2N}. \tag{3.26}$$

Let us focus now on the integral term from Equation (3.20). Introducing the two previous equations leads to

$$\int \frac{2\mathrm{E}[\psi]}{\mathrm{Var}[\psi]} = \int \left(\frac{2N}{\overline{f}}\frac{d\overline{f}}{dp} + 4N\cdot\mathrm{mut}(A_2,A_1)\,p^{-1} - 4N\cdot\mathrm{mut}(A_1,A_2)\,(1-p)^{-1} \right) dp,$$

it can be shown that the solution to this integral is

$$\int \frac{2\mathrm{E}[\psi]}{\mathrm{Var}[\psi]} = 2N\ln(\overline{f}) + 4N\cdot\mathrm{mut}(A_2,A_1)\ln p + 4N\cdot\mathrm{mut}(A_1,A_2)\ln(1-p) + c,$$

where c is an integration constant. Introducing this back into Equation (3.20) yields

$$\phi_{\mathrm{eq}} \propto \overline{f}^{2N}\cdot p^{4N\mathrm{mut}(A_2,A_1)-1}\cdot(1-p)^{4N\mathrm{mut}(A_1,A_2)-1}. \tag{3.27}$$

If one wants to compute the constant hidden in the "proportional to" symbol, we can use the fact that $\phi_{\mathrm{eq}}(p)$ is a probability distribution and therefore $\int_0^1 \phi_{\mathrm{eq}}(p)dp = 1$.

Equation (3.27) is very important since it allows to mathematically analyse many scenarios. For example, Kimura (1983) in his *neutral theory of molecular* considered the absence of selection ($\overline{f}$ constant), to study how much neutral variation is maintained due to the joint application of mutation and genetic drift. In this case, Equation (3.27) simplifies to

$$\phi_{\mathrm{eq}} \propto p^{4N\mathrm{mut}(A_2,A_1)-1} \cdot (1-p)^{4N\mathrm{mut}(A_1,A_2)-1}. \tag{3.28}$$

However, we are more interested in the study of alleles that have a selection disadvantage. This is an important question since, as we saw in previous sections, non-optimal alleles prevent the population from reaching its optimal fitness. Furthermore, we saw in the model depicted by Equation (3.10), that these alleles can reach an equilibrium frequency which might be greater than zero, Equation (3.14). Now we use a similar model where, for simplicity, we no longer consider dominance relationships.

$$\begin{aligned} f_{11} &= 1+\Delta f \\ f_{12} &= 1+\Delta f \\ f_{22} &= 1. \end{aligned} \tag{3.29}$$

Hence, the mean population fitness is $\overline{f} = 1+2\Delta f p - p^2\Delta f$. As discussed earlier, we can neglect terms with p^2, but we can also approximate $e^{2\Delta f p} \approx 1+2\Delta f p$ yielding

$$\overline{f} \approx e^{2\Delta f p}. \tag{3.30}$$

Introducing this back into Equation (3.27) leads to

$$\phi_{\mathrm{eq}} \propto e^{4N\Delta f p} \cdot p^{4N\mathrm{mut}(A_2,A_1)-1} \cdot (1-p)^{4N\mathrm{mut}(A_1,A_2)-1}. \tag{3.31}$$

If we assume that the mutation rates are unbiased, i.e., $\mathrm{mut}(A_1,A_2) = \mathrm{mut}(A_2,A_1) = \mu$, the above expression simplifies to

$$\phi_{\mathrm{eq}} \propto e^{4N\Delta f p} \cdot (p(1-p))^{4N\mu-1}. \tag{3.32}$$

Finally, we plot Equation (3.32) for a relatively recessive allele $\Delta f = 0.01$ with a mutation rate $\mu = 0.00001$ (same as in Figure 3.2). It is really interesting to observe the small peak at $p = 0$ in the blue curve of Figure 3.3. This means that, even though the allele A_2 is selected against, it may be the case that it takes over the whole population! We saw in Subsection 3.1.2 how detrimental alleles prevent the population from reaching the optimal fitness, however here, it can even be the case that the fitness is the minimal possible. On the other hand, the red curve suggests that for bigger population sizes, the peak disappears.

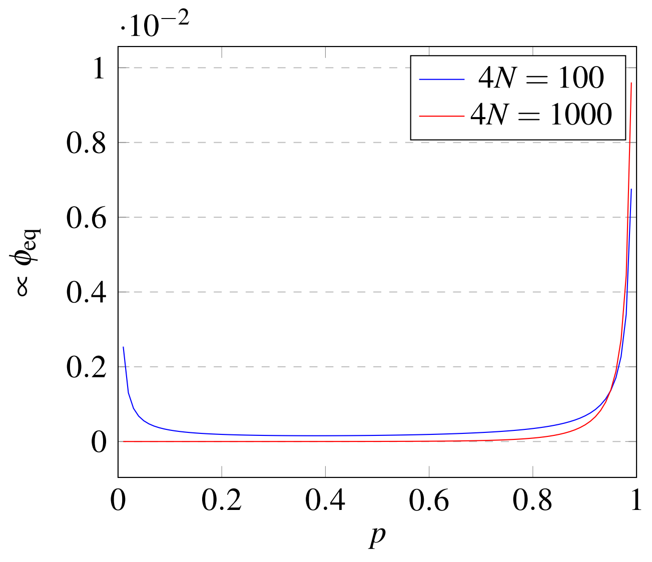


Fig. 3.3 Stationary distribution ϕ_{eq} of the allele A_1 frequencies p from Equation (3.32) with $\mu = 0.00001$. The genotypic fitness model used considers that the allele A_2 is detrimental: $f_{11} = f_{12} = 1.01$ and $f_{22} = 1$. For aesthetic reasons the vertical axis is not normalised.

3.2.2 Probability of Fixation

We have seen the importance of studying detrimental alleles. They cannot only prevent a population to be at the optimal fitness, but also, the population at equilibrium can be composed solely by homozygotes of the detrimental allele. On the other hand, the red curve from Figure (3.3) suggested that the probability of such events occurring might decrease when the population size increases. The straightforward research question is to further investigate this probability, this work was carried on by Kimura (1962) who derived a neat formula such probability.

Going back to Equation (3.31), we can observe another important insight: the probability masses concentrates in both extremes cases of p (see also Figure (3.3)). This is the case when the population consists solely of copies of either the detrimental $p = 0$ or the beneficial allele $p = 1$. When a population is composed solely of one allele we say that such allele goes to *fixation*. Analogously, we write p_{fix} and say *fixation probability* for the probability of the above event occurring. Or mathematically speaking,

$$p_{\mathrm{fix}}(p_0) := \lim_{t \to \infty} \phi(p, t \mid p_0). \tag{3.33}$$

We can now recover Equation (3.24) (the *partial* solution of the Kolmogorov backward equation) and add the boundary condition for the considered regime: $p_{\mathrm{fix}}(1) = 1$ and $p_{\mathrm{fix}}(0) =$

0. Which leads to the expression found by Kimura (1962)

$$p_{\text{fix}}(p_0) = \frac{\int_0^{p_0} e^{-\int \frac{2\text{E}[\psi]}{\text{Var}[\psi]} dp} dp}{\int_0^1 e^{-\int \frac{2\text{E}[\psi]}{\text{Var}[\psi]} dp} dp}. \tag{3.34}$$

We notice immediately that this is a quite complex expression. Again, we can lose some generality and focus on a more specific model. Since we are concerned about the probability of a deleterious allele taking over the whole population, we recover the genotypic fitnesses from Equation (3.29): $f_{11} = 1 + \Delta f$, $f_{12} = 1 + \Delta f$ and $f_{22} = 1$. This model, at H-W frequencies, has a mean population fitness of $\overline{f} = 1 + 2\Delta f p - p^2 \Delta f \approx 1 + 2\Delta f p$. Moreover, using Equation (3.7) (Wright's adaptive landscape) we can estimate the contribution of the directional processes as

$$\text{E}[\psi] = \frac{p(1-p)}{2\overline{f}} \frac{d\overline{f}}{dp} \approx \Delta f p(1-p), \tag{3.35}$$

where we have neglected the effect of mutations since they are very small. As in the previous subsection, the effects of the genetic drift are estimated from the Equation (3.16) (Wright-Fisher model).

$$\text{Var}[\psi] = \frac{p(1-p)}{2N}.$$

Introducing these two equations back into Equation (3.34) leads to

$$p_{\text{fix}}(p_0) = \frac{\int_0^{p_0} e^{-\int 4N\Delta f dp} dp}{\int_0^1 e^{-\int 4N\Delta f dp} dp} = \frac{\int_0^{p_0} e^{-4N\Delta f p} dp}{\int_0^1 e^{-4N\Delta f p} dp} = \frac{1 - e^{4Np_0\Delta f}}{1 - e^{4N\Delta f}}.$$

This formula, first derived by Kimura (1962), provides a simple expression for the probability that an allele that starts with a frequency p_0 will eventually take over the whole population. Conversely, $1 - p_{\text{fix}}(p_0)$ will be the probability of such an allele going to extinction. If we want to consider the above expression when $\Delta f = 0$, we should redefine p_{fix} as follows: $p_{\text{fix}}(p_0, \Delta f = 0) := \lim_{\Delta f \to 0} p_{\text{fix}}(p_0) = p_0$.

We conclude with a graphical representation of Equation (3.36). We observe how it is monotonically increasing with Δf, meaning that the bigger the selection pressure, the smaller the probability that a deleterious allele fixates ($\Delta f < 0$) and the bigger the probability that a beneficial allele fixates ($\Delta f > 0$). Finally, it is interesting to notice that an increase in the population size N, although it decreases p_{fix} no matter the value of Δf, it increases the ratio $p_{\text{fix}}(\Delta f)/p_{\text{fix}}(-\Delta f)$. In other words, the population size highly affects the probability of deleterious alleles fixating, while it does not disrupt too much the fixation probability of beneficial alleles.

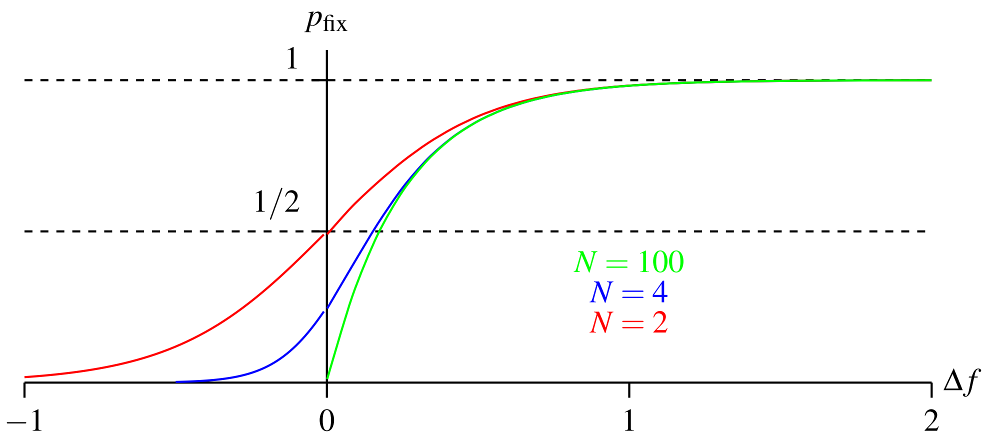


Fig. 3.4 Fixation probability with $p_0 = 1/N$ for $N = 2$ (red line), $N = 10$ (blue line) and $N = 100$ (green line). The horizontal axis represents the fitness difference between the optimal and the minimal genotype.

3.2.3 The Strong Selection Weak Mutation Regime

While tracking how the allele frequencies change over time, we found the importance of studying deleterious alleles. Ultimately, we derived a simple mathematical expression for the fixation probability of any allele. However, in our path to derive this formula, we had to perform the following approximations and assumptions:

- Hardy-Weinberg frequencies (see Definition 3.1).
- Generations are no-overlapping $\Rightarrow$ allows to define fitness in terms of the reproductive contribution per generation (see Definition 3.2).
- Genotypic fitness is independent of the allele frequencies $\Rightarrow$ Wright's adaptive landscape equation holds (see Equation (3.7)).
- Genetic drift is due to random sampling from a finite population $\Rightarrow$ Wright-Fisher model holds (see Equation (3.16)).
- **Weak Mutation:** Mutation rates are very low $\Rightarrow$ avoids to add the mutation rates in the Wright's adaptive landscape equation (see Equation (3.35)).
- **Strong Selection:** Selection is strong enough such that the population goes to fixation before new mutations occur (see Figure 3.5).

The above set of conditions is better known as the *strong selection weak mutation* regime. Because the time to fixation is much smaller than the occurrences of new mutations, under the SSWM regime, the population is nearly always composed of a unique genotype.

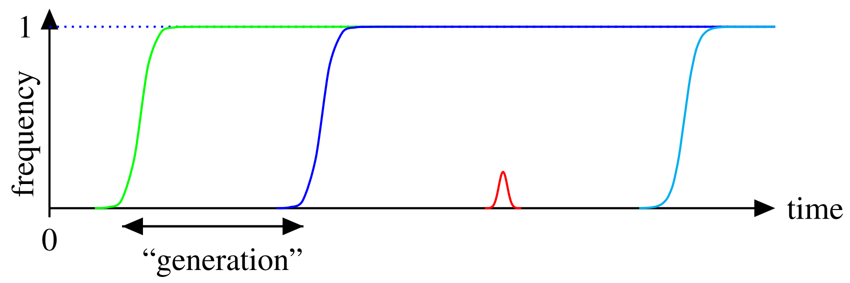


Fig. 3.5 Illustration of the SSWM regime. The y-axis represents the frequency of genotypes that carry a specific mutation. Most of the time the population is composed of a single genotype, as new mutations (represented by different colors) are quickly either fixed (green, blue, cyan) or lost (red).

We also saw that despite the fact that this regime does not constitute a rigorous model (Gillespie, 1989), the conditions are biologically reasonable and the SSWM regime can provide good results. Specially, when the mutation rates are no bigger than $1/(2N)$ and we scale time in units of $2N$ (expected waiting time for a mutation), Karlin and McGregor (1964) proved that the model tends to be exact as N tends to infinity.

Part II

Runtime Analysis of a Natural Evolutionary Regime

Chapter 4

Runtime Analysis and the Speed of Adaption

This chapter is based on the following publications:

1. Paixão, T., Pérez Heredia, J., Sudholt, D., and Trubenová, B. (2015). First steps towards a runtime comparison of natural and artificial evolution. *Proceedings of the Genetic and Evolutionary Computation Conference 2015 (GECCO '15)*, pages 1455–1462. ACM.

2. Paixão, T., Pérez Heredia, J., Sudholt, D., and Trubenová, B. (2017). Towards a runtime comparison of natural and artificial evolution. *Algorithmica*, 78(2):681–713.

3. Pérez Heredia, J., Trubenová, B., Sudholt, D., and Paixão, T. (2017). Selection limits to adaptive walks on correlated landscapes. *Genetics*, 205(2):803–825, 2017.

In this chapter we discuss one of the most important topics in population genetics: *the speed of adaptation*. After motivating and introducing the problem, we will discuss the state of the art and how our proposed approach can improve it. As pointed out in the introduction, we will exploit the similarities between EC and PG to argue that runtime analysis is an equivalent research question to the study of the speed of adaptation. However, there is a discrepancy in the nomenclature that both fields use to denote equivalent concepts. In Table 4.1 we present an informal translation of some relevant terms that will be used in subsequent chapters. For a further translation and unified framework we refer the interested reader to Paixão et al. (2015), from where Table 4.1 is inspired.

PG	EC	**Meaning**
Genotype	Solution	Where natural or artificial evolution takes place.
Locus	Bit	Each component of a genotype or solution.
Allele $\in \{A,C,G,T\}$	Bit-value $\in \{0,1\}$	Possible values of a locus or a bit.
Number of sites under selection m	Problem size n	The (relevant) length of a genotype or solution.
Genotype space $\{A,C,G,T\}^m$	Search space $\{0,1\}^n$	The set from where genotypes or solution belong to.
Adaptive walk	Trajectory	A collection during time of a genotype (solution) and its offspring genotypes.
Genetic drift	Genetic drift	Stochasticity associated with sampling from finite populations.
-	Drift	Expected change of a stochastic process.
Fitness	Fitness	**PG:** Reproductive contribution of an individual to the next generation. **EC:** Measure of how good a solution is.
Genotypic Fitness	-	Average fitness across individuals with the same genotype in a population.
Speed of adaptation	Runtime	How quickly a natural population evolves complex adaptations. This is equivalent to the first hitting time of the optimum in an optimisation problem.
Mutational effects	Bit weights	Fitness contribution of a bit-position (locus) being set to a certain value (allele).

Table 4.1 Translation of some concepts between PG and EC.

The question of how long it takes for a natural population to evolve complex adaptations has fascinated researchers for decades (Grant and Flake, 1974; Haldane, 1957; Kimura, 1961; Valiant, 2013). The evolution of populations can be seen as an adaptive walk (trajectories) across the “mutational landscape” (search space), the space of all possible genotypes (Gillespie, 1984). The speed of adaptation (runtime) critically depends on how the fitness values of all genotypes (solutions) are organised in this space (fitness landscape). In particular, it depends on the number and shape of the fitness paths leading to the optimum on this landscape. This raises both empirical and theoretical difficulties for the study of the speed of adaptation. Empirically, measuring the fitness of every possible genotype is virtually impossible. For this reason, most empirical studies focused on distributions of effects of single mutants (Eyre-Walker and Keightley, 2007). However, organisms are not just the sum of their genes: gene interactions (epistasis) are pervasive and the effects of mutations will change depending on the background they occur in (Phillips, 2008). The difficulty of measuring mutational effects across multiple backgrounds grows combinatorially with the length of the genotype and most studies are restricted to studying the effects of interactions in a local neighbourhood of some genotype. In part because of this lack of knowledge about the structure of the fitness landscape, and in part due to the added difficulty of analysing correlated landscapes, most theoretical studies have focused on landscapes in which either the fitness of genotypes (Gillespie, 1983, 1984; Kauffman and Levin, 1987; Orr, 2002) or the effects of new mutations (Desai et al., 2007; Fogle et al., 2008; Wilke, 2004) are drawn from a random distribution. The first case, adaptation on random landscapes, leads to extremely short adaptive walks and may be realistic only when the population is very close to a fitness peak (Orr, 2006). In the second case, adaptation in linear landscapes, such as when the effects of mutations are drawn from a random distribution, ignores potential correlations between mutational neighbourhoods and any kind of interaction between mutations.

Most studies on the speed of adaptation have focused on the limits imposed by competition between multiple beneficial mutations (Gerrish and Lenski, 1998). Because of this, most models assume that populations evolve in a continuous space under a never ending supply of beneficial mutations ((Orr, 2000; Wilke, 2004), but see Kim and Orr (2005) for a model of a finite genome), when in reality the stage in which evolution proceeds is composed of discrete genotypes. This fact results in a number of new and important features for the dynamics of adaptation. First, in a discrete space of genotypes the supply of new beneficial mutations naturally decreases as adaptation occurs, as a consequence of the finite size of the genome. Second, and consequently, as the population becomes more adapted the potential for deleterious mutations increases as more and more sites become adapted. Models analysing adaptive walks typically assume that the population or selection strength are large enough such that the probability of fixation of deleterious mutations is zero, effectively disregarding the growing difficulty of maintaining the acquired adaptations. Finally, fitness landscapes can display strong correlations between mutational neighbourhoods, making the effects of new mutations not necessarily constant across the fitness landscape nor simply drawn from a random distribution. Previous attempts at analysing the speed of adaptation in correlated neighbourhoods (Kryazhimskiy et al., 2009) assumed an infinite supply of beneficial mutations and strong selection, disregarding the growing difficulty of finding new beneficial mutations and maintaining previously acquired ones. As we will show, these effects impose strong constraints to adaptation.

Other studies have focused on properties of adaptive walks, which explicitly consider the discrete nature of the genotype space (Kauffman and Levin, 1987; Orr, 2002; Park et al., 2016). Many of these studies have focused on models of fitness landscapes that can display high levels of ruggedness, such as the House-of-Cards model (Kingman, 1978), in which *fitness values* are drawn randomly from some distribution, the Rough Mount Fuji (Aita et al., 2000), in which *fitness effects*, combined with a deterministic part of fitness, are drawn randomly, or the NK-model in which the fitness effect of a locus depends in some randomly prescribed way on the state of K other loci (Kauffman and Weinberger, 1989). Both of these classes of models lead to landscapes exhibiting multiple peaks. For this reason, these studies have focused mainly on the length of the adaptive walk, the number of substitutions that occurs before the process reaches a local peak, and how this depends on the number of local peaks in the landscape. Even though this is an empirically measurable quantity it does not directly address the question of how long a population takes to reach this peak and how this depends on the shape of the paths leading up to it. Note that the number of substitutions is not equivalent to the time it takes to reach a peak: new mutations, even if beneficial, can be lost, and deleterious mutations can be fixed. Here, we directly address this question by asking how much time a population requires to reach a fitness peak.

In order to do this, instead of considering the rate of adaptation in specific fitness landscapes, which may not be informative of real trajectories since their details are unknowable, we consider *classes* of fitness landscapes, including many patterns of gene interactions, and focus on upper bounds for the time to reach a fitness peak. We focus on traits encoded by many genes and study how this time depends on the number of sites under selection (problem size). We argue that the scaling of this time with the length of the target sequence quantifies the complexity or "hardness" for a natural population to perform an adaptive walk on a class of landscapes. Similar to previous approaches (Gillespie, 1983, 1984; Orr, 2002, 2005, 2006) we consider a monomorphic population in the weak mutation regime. However, in order to address the difficulties outlined above, we consider that this population evolves in a sequence space and under the combined action of mutation, selection, and drift, allowing for the possibility that deleterious mutations are fixed.

In order to analyse the dynamical properties of the adaptive trajectory we take advantage of tools commonly used in the theory of randomised and evolutionary algorithms (see Chapter 2). Interestingly, although evolutionary algorithms are heavily inspired by natural evolution, these methods have seldom been applied to natural evolution as studied in mathematical population genetics. This is a missed opportunity: the time it takes for a natural population to reach a fitness peak is an important question for the study of natural evolution. The kinds of results obtained from runtime analysis, namely how the runtime scales with genome size and mutation rate, are of general interest to population genetics. Moreover, recently there has been a renewed interest in applying computer science methods to problems in evolutionary biology with contributions from unlikely fields such as game theory (Chastain et al., 2014), machine learning (Valiant, 2009), Markov chain theory (Chatterjee et al., 2014) or even the more general aim of unifying both fields (Paixão et al., 2015).

In this part of the thesis, we present an application of runtime analysis to the so-called Strong Selection Weak Mutation regime of natural populations (see Subsection 3.2.3). However, in order to apply the runtime analysis techniques introduced in Chapter 2, we must establish a relationship between the SSWM regime and evolutionary algorithms.

4.1 SSWM as a Trajectory-Based Algorithm

In Chapter 3, we saw that: the SSWM model applies when mutations are rare enough and selection is strong enough that the time between occurrences of new mutations is long compared to the time a new genotype takes to replace the parent genotype. Mutations occur rarely either because the mutation rate is low, or because the size of the underlying population is small. Upon occurrence, a new mutation represents relatively high fitness advantage or fitness loss, and strong selection ensures that it either promptly replaces the original genotype, or is entirely lost from the population (Figure 3.5). Therefore, the population is most of the time composed of a single genotype and evolution occurs through “jumps” between different genotypes, corresponding to a new mutated genotype replacing the resident genotype in the population. Which one of the two genotypes is the survivor is determined by the so-called fixation probability (see Equation (3.36)). Hence, the relevant dynamics can then be characterised by a Markov process evolving just one genotype.

As can be seen from the description above, the model resembles a trajectory-based algorithm (see Algorithm 2.2). However, for a meaningful translation we must keep in mind the assumptions and approximations described in Subsection 3.2.3. Specially, we should comply with the guidelines by Karlin and McGregor (1964), therefore we will use local mutations and we will measure time as the number of mutations. To recover the real time it suffices to multiply by the expected waiting time for a mutation. Finally, it is important to note that the derivations from Chapter 3 considered a diploid population of size N ($2N$ locus), however in Chapter 2 we were using an haploid encoding. Fortunately, to translate our results for the haploid scenario, it suffices to exchange $2N \leftrightarrow N$ in the expression of the fixation probability (see Equation (3.36)). Hence,

$$p_{\text{fix}}(\Delta f) = \frac{1 - e^{-2\beta\Delta f}}{1 - e^{-2N\beta\Delta f}}, \tag{4.1}$$

where we have used $p_0 = 1/N$ since we are using local mutations and we have introduced a new parameter $\beta \in \mathbb{R}^+$ which acts as a scaling factor. In the next section we will deeply analyse this expression, but first we conclude the process of casting the SSWM regime as an evolutionary algorithm. By using this fixation probability as the acceptance function in Algorithm 2.3, we finally obtain the pseudo-code for the SSWM algorithm.

Algorithm 4.1: SSWM

```
1  Initialise $x \in \{0,1\}^n$
2  Choose a population size $N \in \mathbb{N}$
3  Choose a scaling parameter $\beta \in \mathbb{R}^+$
4  repeat
5      $y \leftarrow$ flip a uniform randomly chosen bit from $x$
6      $\Delta f \leftarrow f(y) - f(x)$
7      Choose $r \in [0,1]$ uniformly at random
8      if $r \leq p_{\text{fix}}(\Delta f)$ then
9          $x \leftarrow y$
10 until stop;
```

Our analysis of the SSWM algorithm in subsequent chapters, will not only be meaningful for the PG community, but will also improve the understanding of non-elitist EAs which are much less studied than the elitist versions. The use of p_{fix} introduces two main differences to the (1+1) EA (Algorithm 2.6). First, while the (1+1) EA never decreases its current fitness (a property called *elitism*), SSWM accepts solutions of lower fitness (worsenings) with some positive probability. This is reminiscent of the Metropolis algorithm (Algorithm 2.7) which can also accept worsenings (see e. g. Metropolis et al., 1953). Second, and in contrast to the Metropolis algorithm, solutions of higher fitness can be rejected, since they are accepted with a probability that is roughly proportional to the relative advantage they have over the current solution.

The main technical difficulties are that in contrast to the simple (1+1) EA, SSWM is a non-elitist algorithm, hence fitness-level arguments based on elitism are not applicable. In a nutshell, elitist fitness-level methods break the search space into m partitions according to increasing fitness values. Then, if the algorithm is elitist, the expected runtime can be upper bounded by $\sum_{i=1}^{m} 1/p_i$, where p_i is a lower bound on the probability of sampling a search point in a partition of higher fitness than the current partition (see e.g. Oliveto and Yao, 2011). Although fitness-levels have been extended to consider non-elitism (Corus et al., 2014), it is required that the algorithm has a population size larger than 1 (which is not the case of SSWM). Moreover, while for the (1+1) EA transition probabilities to better solutions are solely determined by probabilities for flipping bits during mutation, for SSWM these additionally depend on the fitness difference. The analysis of SSWM is more challenging than the analysis of the (1+1) EA, and requires tailored proof techniques.

4.2 Understanding the Fixation Probability

Before starting to analyse the SSWM regime as an algorithm, it is important to improve our understanding of the fixation probability. This is crucial since p_{fix} is the characteristic that differentiates SSWM from other trajectory-based algorithms. We start by studying the meaning of the parameters of the fixation probability: Δf, N and β. For the sake of readability we present again the expression for the fixation probability.

$$p_{\text{fix}}(\Delta f) = \frac{1 - e^{-2\beta\Delta f}}{1 - e^{-2N\beta\Delta f}}, \tag{4.2}$$

where $\Delta f \neq 0$ is the fitness difference to the current solution and $N \geq 1$ is the size of the underlying population (see Chapter 3). From an algorithmic point of view N is just a parameter. For $\Delta f = 0$ we define $p_{\text{fix}}(0) := \lim_{\Delta f \to 0} p_{\text{fix}}(\Delta f) = \frac{1}{N}$, so that p_{fix} is continuous and well defined for all Δf. If $N = 1$, this probability is $p_{\text{fix}}(\Delta f) = 1$, meaning that any offspring will be accepted, and if $N \to \infty$, it will only accept solutions for which $\Delta f > 0$.

To finalise studying the effect of the population size N on the fixation probability, we show that for all $N > 1$ the function p_{fix} is strictly increasing.

Lemma 4.1. *p_{fix} is monotonic for all $N \geq 1$ and strictly increasing for $N > 1$.*

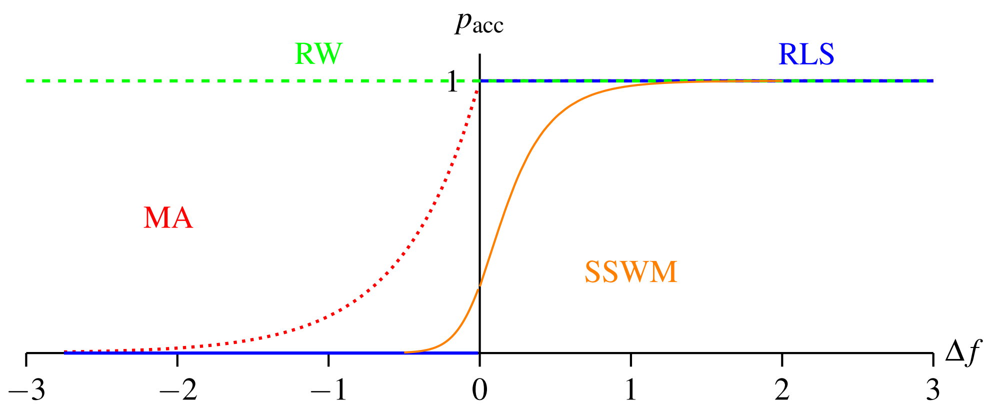


Fig. 4.1 Acceptance probability for RW (green dashed line), RLS and (1+1) EA (blue solid line), MA (red dotted line) and SSWM (orange solid line).

Proof. If $N = 1$, $p_{\text{fix}}(\Delta f) = 1$. In order to show that $p_{\text{fix}}(\Delta f)$ is monotonically increasing for $N > 1$ we show that $\frac{\partial p_{\text{fix}}(\Delta f)}{\partial \Delta f} > 0$ for all $\Delta f \neq 0$. The derivative is

$$\frac{\partial p_{\text{fix}}(\Delta f)}{\partial \Delta f} = \frac{2\beta \cdot e^{-2\Delta f}}{1 - e^{-2N\Delta f}} - \frac{2N\beta \cdot e^{-2N\Delta f}(1 - e^{-2\Delta f})}{(1 - e^{-2N\Delta f})^2}.$$

Dividing the derivative by 2β and multiplying by $(1 - e^{-2N\Delta f})^2$, which for $\Delta f \neq 0$ is always positive, we get

$$\begin{aligned}\frac{(1 - e^{-2N\Delta f})^2}{2\beta} \frac{\partial p_{\text{fix}}(\Delta f)}{\partial \Delta f} &= e^{-2\Delta f}\left(1 - e^{-2N\Delta f}\right) - N \cdot e^{-2N\Delta f}\left(1 - e^{-2\Delta f}\right) \\ &= e^{-2(N+1)\Delta f}\left(e^{2N\Delta f} + N - 1 - Ne^{2\Delta f}\right).\end{aligned}$$

Hence $\frac{\partial p_{\text{fix}}(\Delta f)}{\partial \Delta f} > 0$ if and only if

$$e^{2N\Delta f} + N - 1 > Ne^{2\Delta f}. \tag{4.3}$$

If $N = 1$ then $e^{2N\Delta f} + N - 1 = Ne^{2\Delta f}$, and from comparing the derivatives w. r. t. N of both sides of (4.3):

$$\frac{\partial}{\partial N}\left(e^{2N\Delta f} + N - 1\right) = 2\Delta f e^{2N\Delta f} + 1 \tag{4.4}$$

$$\frac{\partial}{\partial N} Ne^{2\Delta f} = e^{2\Delta f} \tag{4.5}$$

we see that for $N = 1$ the derivative (4.4) is larger than derivative (4.5). Since expression (4.4) is an increasing positive function of N for any Δf while expression (4.5) is constant in N, the inequality (4.3) is established for $N > 1$ and hence $\frac{\partial p_{\text{fix}}(\Delta f)}{\partial \Delta f} > 0$. $\square$

In Figure 4.2 we observe the same behaviour as in Figure 3.4, increasing N rapidly reduces the probability of accepting both detrimental and slightly beneficial mutations. However, for beneficial enough mutations, the value of N does not yield a significant change in p_{fix}.

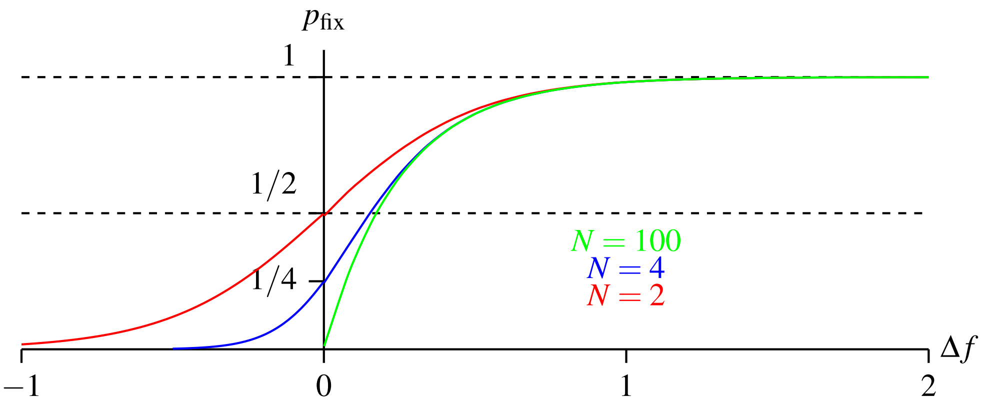


Fig. 4.2 Fixation probability with $\beta = 2$ and $N = 2$ (red), $N = 10$ (blue) and $N = 100$ (green).

We now move on to the study of the parameter β. The reason behind introducing this parameter is that the acceptance function in this algorithm depends on the difference in fitness between genotypes. By incorporating β as a parameter of this function (and hence of the algorithm) we avoid having to explicitly rescale the fitness functions we analyse, while allowing us to explore the performance of this algorithm on a family of functions.

As depicted in Figure 4.1, the fixation probability resembles a sigmoidal shape with limits $\lim_{\Delta f \to -\infty} p_{\text{fix}}(\Delta f) = 0$ as well as $\lim_{\Delta f \to \infty} p_{\text{fix}}(\Delta f) = 1$ (Figure 4.1). Similar limits are obtained when β tends to ∞,

$$\lim_{\beta \to \infty} p_{\text{fix}}(\Delta f) = \begin{cases} 0 & \text{if } \Delta f < 0 \\ 1/N & \text{if } \Delta f = 0 \\ 1 & \text{if } \Delta f > 0 \end{cases}$$

As such, for large $|\beta \Delta f|$ this probability of acceptance is close to the one in the (1+1) EA (as long as $N > 1$) defeating the purpose of the comparison, with the only difference being the tie-breaking rule. While the (1+1) EA always accepts the new solution in case of a tie in fitness ($\Delta f = 0$), SSWM only accepts the new solution with probability $1/N$.

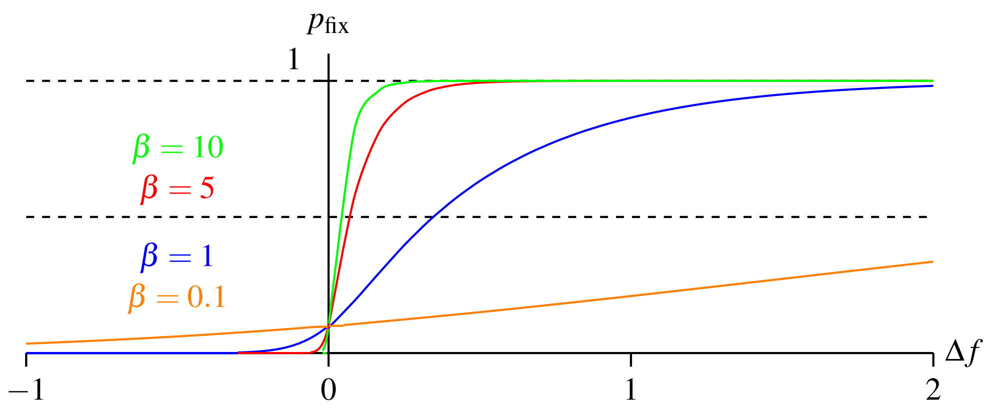


Fig. 4.3 Fixation probability with $N = 10$ and $\beta = 0.1$ (orange), $\beta = 1$ (blue), $\beta = 5$ (red) and $\beta = 10$ (green).

After improving our understanding of the parameters β and N, we return to a general study of the fixation probability. We conclude this chapter presenting three powerful lemmas that, in addition to improving the understanding of p_{fix}, will simplify the analysis of the SSWM algorithm. First, the following lemma derives upper and lower bounds for $p_{\text{fix}}(\Delta f)$.

Lemma 4.2. *For every $\beta \in \mathbb{R}^+$ and $N \in \mathbb{N}^+$ the following inequalities hold. If $\Delta f > 0$ then*

$$\frac{2\beta\Delta f}{1+2\beta\Delta f} \leq p_{\text{fix}}(\Delta f) \leq \frac{2\beta\Delta f}{1-e^{-2N\beta\Delta f}}.$$

If $\Delta f < 0$ then

$$\frac{-2\beta\Delta f}{e^{-2N\beta\Delta f}} \leq p_{\text{fix}}(\Delta f) \leq \frac{e^{-2\beta\Delta f}}{e^{-2N\beta\Delta f}-1}.$$

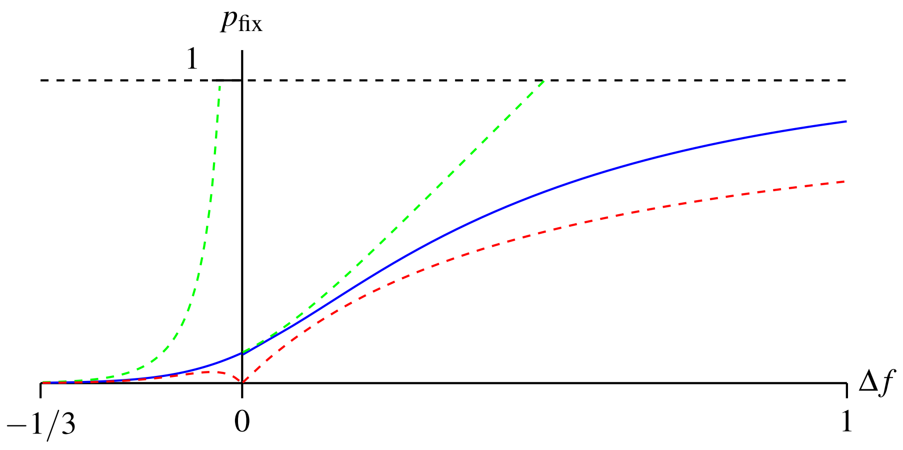


Fig. 4.4 Fixation probability with $N = 10$ and $\beta = 1$ (blue line) and lower (red dashed line) and upper (green dashed line) bounds from Lemma 4.2.

Proof of Lemma 4.2. In the following we frequently use $1+x \le e^x$ and $1-e^{-x} \le 1$ for all $x \in \mathbb{R}$ as well as $e^x \le \frac{1}{1-x}$ for $x<1$. If $\Delta f > 0$,

$$p_{\text{fix}}(\Delta f) = \frac{1-e^{-2\beta\Delta f}}{1-e^{-2N\beta\Delta f}} \ge 1-e^{-2\beta\Delta f} \ge 1-\frac{1}{1+2\beta\Delta f} = \frac{2\beta\Delta f}{1+2\beta\Delta f}$$

as well as

$$p_{\text{fix}}(\Delta f) \;=\; \frac{1-e^{-2\beta\Delta f}}{1-e^{-2N\beta\Delta f}} \le \frac{2\beta\Delta f}{1-e^{-2N\beta\Delta f}}.$$

If $\Delta f < 0$, using the fact that $e^{-x}-1 \le e^{-x}$:

$$p_{\text{fix}}(\Delta f) \;=\; \frac{e^{-2\beta\Delta f}-1}{e^{-2N\beta\Delta f}-1} \le \frac{e^{-2\beta\Delta f}}{e^{-2N\beta\Delta f}-1}.$$

Similarly:

$$p_{\text{fix}}(\Delta f) \;=\; \frac{e^{-2\beta\Delta f}-1}{e^{-2N\beta\Delta f}-1} \ge \frac{e^{-2\beta\Delta f}-1}{e^{-2N\beta\Delta f}} \ge \frac{-2\beta\Delta f}{e^{-2N\beta\Delta f}}. \quad \square$$

The next lemma shows that the probability of accepting an improvement of Δf is exponentially larger (in $2(N-1)\beta\Delta f$) than accepting its symmetric fitness variation $-\Delta f$.

Lemma 4.3. *For every $\beta \in \mathbb{R}^+$, $\Delta f \in \mathbb{R}$ and $N \in \mathbb{N}^+$*

$$\frac{p_{\text{fix}}(-\Delta f)}{p_{\text{fix}}(+\Delta f)} = e^{-2(N-1)\beta\Delta f}.$$

Proof.

$$\begin{aligned}\frac{p_{\text{fix}}(-\Delta f)}{p_{\text{fix}}(+\Delta f)} &= \frac{e^{2\beta\Delta f}-1}{e^{2N\beta\Delta f}-1}\cdot\frac{1-e^{-2N\beta\Delta f}}{1-e^{-2\beta\Delta f}} \\ &= \frac{e^{2\beta\Delta f}}{e^{2N\beta\Delta f}} = e^{-2(N-1)\beta\Delta f}\end{aligned}$$

where we have applied the relation $\frac{e^x-1}{1-e^{-x}} = e^x$. $\square$

Finally, we mathematically express the rapid decreasing with Δf of the probability of accepting a worsening. The next lemma shows that: when comparing two deleterious mutations with a fitness difference of $\delta \in \mathbb{R}^+$ between them, the probability of accepting the worse mutation $p_{\text{fix}}(\Delta f - \delta)$ is exponentially smaller (in δ) than the probability of accepting the best, but still deleterious mutation, $p_{\text{fix}}(\Delta f)$.

Lemma 4.4. *Let $\delta \in \mathbb{R}^+$, $\beta \in \mathbb{R}^+$ and $\Delta f \leq 0$, then p_{fix} decreases exponentially with δ*

$$p_{\text{fix}}(\Delta f) \geq e^{\delta} \cdot p_{\text{fix}}(\Delta f - \delta)$$

provided

$$2\beta(N-1) \geq 1 + 2 \cdot \max\left(1, \frac{1}{\delta}\right).$$

Proof.

$$\begin{aligned}
& e^{\delta} \cdot p_{\text{fix}}(\Delta f - \delta) \leq p_{\text{fix}}(\Delta f) \\
\Leftrightarrow\ & e^{\delta} \cdot \frac{e^{-2\beta(\Delta f - \delta)} - 1}{e^{-2\beta N(\Delta f - \delta)} - 1} \leq \frac{e^{-2\beta \Delta f} - 1}{e^{-2\beta N \Delta f} - 1}
\end{aligned}$$

defining $x := e^{-2\beta\Delta f}$ and $y := e^{-2\beta\delta}$

$$\begin{aligned}
\Leftrightarrow\ & e^{\delta} \cdot \frac{\frac{x}{y} - 1}{\left(\frac{x}{y}\right)^N - 1} \leq \frac{x-1}{x^N - 1} \\
\Leftrightarrow\ & e^{\delta} \cdot \frac{1}{\sum_{k=0}^{N-1} \left(\frac{x}{y}\right)^k} \leq \frac{1}{\sum_{k=0}^{N-1} x^k} \\
\Leftrightarrow\ & e^{\delta} \cdot \sum_{k=0}^{N-1} x^k \leq \sum_{k=0}^{N-1} \left(\frac{x}{y}\right)^k.
\end{aligned}$$

Splitting the sums over the point $2\beta k = 1$ yields

$$\Leftrightarrow \sum_{k=0}^{\left\lfloor \frac{1}{2\beta} \right\rfloor} e^{\delta} \cdot x^k + \sum_{k=\left\lceil \frac{1}{2\beta} \right\rceil}^{N-1} e^{\delta} \cdot x^k \leq \sum_{k=0}^{\left\lfloor \frac{1}{2\beta} \right\rfloor} \left(\frac{x}{y}\right)^k + \sum_{k=\left\lceil \frac{1}{2\beta} \right\rceil}^{N-1} \left(\frac{x}{y}\right)^k.$$

Grouping the terms with the same summation limits and recovering $y := e^{-2\beta\delta}$ yields

$$\Leftrightarrow \sum_{k=0}^{\left\lfloor \frac{1}{2\beta} \right\rfloor} \left(e^{\delta} - e^{2\beta\delta k}\right) x^k \leq \sum_{k=\left\lceil \frac{1}{2\beta} \right\rceil}^{N-1} \left(e^{2\beta\delta k} - e^{\delta}\right) x^k$$

and by using $e^{\delta} - e^{2\beta\delta k} \leq e^{\delta}$ we get

$$\Leftarrow \sum_{k=0}^{\left\lfloor \frac{1}{2\beta} \right\rfloor} e^{\delta} x^k \leq \sum_{k=\left\lceil \frac{1}{2\beta} \right\rceil}^{N-1} \left(e^{2\beta\delta k} - e^{\delta}\right) x^k. \tag{4.6}$$

We need to find the point from where $e^{2\beta\delta k} - e^{\delta} > e^{\delta}$

$$e^{2\beta\delta k} > 2e^{\delta} \Leftrightarrow \delta(2\beta k - 1) > \ln 2 \Leftrightarrow k\beta > \frac{\ln 2}{\delta} + 1.$$

Separating δ greater and smaller than 1 leads to

$$\Leftarrow k \geq \max\left(\frac{1}{\beta}, \frac{1}{\delta\beta}\right).$$

Going back to the previous calculations (4.6) leads to

$$\begin{aligned}
&\Leftrightarrow \sum_{k=0}^{\left\lfloor \frac{1}{2\beta} \right\rfloor} e^{\delta} x^k \leq \sum_{k=\left\lceil \frac{1}{2\beta} \right\rceil}^{\left\lfloor \max\left(\frac{1}{\beta}, \frac{1}{\delta\beta}\right) \right\rfloor} \left(e^{2\beta\delta k} - e^{\delta}\right) x^k + \sum_{k=\left\lceil \max\left(\frac{1}{\beta}, \frac{1}{\delta\beta}\right) \right\rceil}^{N-1} e^{\delta} x^k \\
&\Leftarrow \sum_{k=0}^{\left\lfloor \frac{1}{2\beta} \right\rfloor} e^{\delta} x^k \leq \sum_{k=\left\lceil \max\left(\frac{1}{\beta}, \frac{1}{\delta\beta}\right) \right\rceil}^{N-1} e^{\delta} x^k.
\end{aligned}$$

Now we are comparing two polynomials with equal positive coefficients, but as the right-hand degrees are greater (note that $x \geq 1$), we get

$$\begin{aligned}
\Leftarrow \sum_{k=0}^{\left\lfloor \frac{1}{2\beta} \right\rfloor} e^{\delta} &\leq \sum_{k=\left\lceil \max\left(\frac{1}{\beta}, \frac{1}{\delta\beta}\right) \right\rceil}^{N-1} e^{\delta} \\
\Leftrightarrow e^{\delta} \cdot \left(\left\lfloor \frac{1}{2\beta} \right\rfloor + 1\right) &\leq e^{\delta} \cdot \left(N - \left\lceil \max\left(\frac{1}{\beta}, \frac{1}{\delta\beta}\right) \right\rceil\right) \\
\Leftarrow \frac{1}{2\beta} &\leq N - \max\left(\frac{1}{\beta}, \frac{1}{\delta\beta}\right) - 1.
\end{aligned}$$

Solving for N yields

$$\Leftrightarrow N \geq \frac{1}{2\beta} + 1 + \max\left(\frac{1}{\beta}, \frac{1}{\delta\beta}\right). \qquad \square$$

4.3 Conclusions

This chapter has established a relationship between two important questions from two different research fields. Whereas in PG the question of how long it takes for a natural population to evolve complex adaptations is of high relevance, in EC we find that researchers are interested in the time that an EA needs to find the optimum of a given function. Since these research questions are equivalent, we propose to apply the runtime analysis techniques introduced in Section 2.3 to study the speed of adaptation. Precisely, as motivated in Chapter 3, we would like to study the speed of adaptation of the SSWM regime.

However, in order to apply the analysis tools from EAs, we first showed that the SSWM regime can be cast as a trajectory-based algorithm (recall Algorithm 4.1). This algorithmic version of the SSWM regime will use as acceptance probability the fixation probability derived by theoretical biologists. Since this is the distinguishing characteristic from other trajectory-based algorithms, we used a full subsection devoted to improve our understanding of the fixation probability.

We derived several lemmas that will be useful in subsequent chapters. Firstly, we showed that $p_{\text{fix}}$is monotonic with respect to Δf, i.e., it never decreases while Δf increases. Secondly, to ease future calculations, we derived simple mathematical expressions that lower and upper bound the fixation's probability value. Then, we found an equality for the quotient of the fixation probability of two symmetrical fitness differences (Δf and $-\Delta f$), the result shows that this ratio is exponentially big in $2(N-1)\beta\Delta f$. Finally, we showed that for two detrimental moves with $\Delta f_1, \Delta f_2 < 0$, the quotient of their p_{fix}values is, at least, exponentially big in $\Delta f_2 - \Delta f_1$.

Chapter 5

Speed of Adaptation in Additive Landscapes

Having laid the foundations in Chapter 4, Chapter 5 is also based on these papers:

1. Paixão, T., Pérez Heredia, J., Sudholt, D., and Trubenová, B. (2015). First steps towards a runtime comparison of natural and artificial evolution. *Proceedings of the Genetic and Evolutionary Computation Conference 2015 (GECCO '15)*, pages 1455–1462. ACM.

2. Paixão, T., Pérez Heredia, J., Sudholt, D., and Trubenová, B. (2017). Towards a runtime comparison of natural and artificial evolution. *Algorithmica*, 78(2):681–713.

3. Pérez Heredia, J., Trubenová, B., Sudholt, D., and Paixão, T. (2017). Selection limits to adaptive walks on correlated landscapes. *Genetics*, 205(2):803–825, 2017.

After casting the SSWM regime as a trajectory-based algorithm in the previous chapter, we are ready to apply the runtime analysis techniques from Chapter 2. Using these tools, we first calculate an upper bound for the time to reach an adaptive peak (fitness peak) in a simple landscape with equal, additive contributions of all sites (loci or bit-positions) as a function of the number of such sites contributing to the trait (problem size). This landscape is formalised by the already introduced problem of ONEMAX.

As discussed in Section 2.3, we focus on the crucial distinction between a polynomial and an exponential scaling of this time with the number of sites under selection, and argue that these two qualitatively distinct regimes correspond to situations in which adaptation is "efficient" or "inefficient", respectively. We find conditions on the selection strength that separate these two regimes, and show that populations in the weak mutation regime can adapt efficiently, but the critical selection strength grows with the number of sites under selection, effectively setting a limit to adaptation. We generalise these results to a large family of fitness landscapes that includes very general forms of interactions between the sites under selection, excluding only forms of interactions that create multiple fitness peaks (this will be analysed in Chapters 6 and 7). We derive an upper limit to the time to reach a fitness peak, setting a lower speed limit to adaptation in these landscapes.

Finally, we analyse in detail one instance of this class, an extreme form of epistasis in which mutations need to be accumulated in a particular order. We show that in this case, despite a slower speed of adaptation, the critical selection strength enabling efficient adaptation does not depend on the number of sites under selection, eliminating the limits to adaptation previously identified for simpler landscapes.

5.1 Simple Hill Climbing Tasks

One of the simplest scenarios for adaptation is when all sites—genes, or loci—contribute equally to the fitness. This leads to a fitness landscape where the fitness of a genotype depends only on the number of correct matches to a target sequence. We formalise this scenario by the function $\text{ONEMAX}(x) = \sum_{i=1}^{n} x_i$ (see Definition 2.4), which counts the number of correct matches x in a genome of length n. This function induces a structure in sequence space in which the fraction of beneficial mutations decreases linearly as a function of the distance to the optimum. We use this function to determine under which conditions populations can efficiently climb simple fitness peaks.

From the algorithmic perspective, the ONEMAX function has been extensively studied because of its simplicity. It represents an easy hill climbing task, and it is the easiest function with a unique optimum for all evolutionary algorithms that only use standard bit mutation for variation (Sudholt, 2013). Showing that SSWM can optimise ONEMAX efficiently serves as proof of concept that SSWM is a reasonable optimiser. It further sheds light on how to set algorithmic parameters such as the selection strength β and the population size N. To this end, we first show a polynomial upper bound for the runtime of SSWM on ONEMAX for a selection strength of $2(N-1)\beta \geq \ln(cn)$. We then show that SSWM exhibits a phase transition on its runtime as a function of $2N\beta$; decreasing this parameter by a constant factor below $\ln n$ leads to exponential runtimes on ONEMAX (see Figure 5.1).

Another reason why studying ONEMAX for SSWM makes sense is because not all evolutionary algorithms that use a fitness-dependent selection perform well on ONEMAX. Neumann et al. (2009) as well as Oliveto and Witt (2014) showed that evolutionary algorithms using fitness-proportional selection, including the Simple Genetic Algorithm, fail badly on ONEMAX even within exponential time, with very high probability.

The following theorem shows that SSWM is efficient on ONEMAX whenever $2(N-1)\beta \geq \ln(cn)$ for some constant $c > 1$, since then $p_{\text{fix}}(1)$ starts being greater than $n \cdot p_{\text{fix}}(-1)$, allowing for a positive drift even on the hardest fitness level ($n-1$ ones). The obtained expression quantifies the impact of the length of the target sequence on the time (in units of mutation rate) to attain it. It shows that the time required to evolve adaptations involving larger numbers of sites will simply require a polynomial number of extra mutational "trials", provided that $\beta = 1/\text{poly}(n)$.

Theorem 5.1. *If $2(N-1)\beta \geq \ln(cn)$ with $\beta \in \mathbb{R}^+$ and $c > 1$, then the expected optimisation time of SSWM on* ONEMAX *is at most*

$$\frac{n\ln(n) + O(n)}{p_{\text{fix}}(1)} \leq \left(1 + \frac{1}{2\beta}\right) \cdot (n\ln(n) + O(n))$$

for every initial search point.

Proof. The drift (see Definition 2.12) can be expressed as a combination of a forward and a backward drift

$$\Delta(x) = \Delta^+(x) - |\Delta^-(x)|$$

where the forward drift is the probability of mutation flipping a 0-bit ($(n-x)/n$) multiplied by the probability of accepting such a mutation ($p_{\text{fix}}(1)$). Note that all mutations in this fitness landscape have fitness contribution of ± 1. Analogously, the backward drift, is given by the probability of negative mutation occurring (x/n) and fixing in the population with probability ($p_{\text{fix}}(-1)$). Therefore, the total expected progress is

$$\begin{aligned} \Delta(x) &= \frac{n-x}{n} \cdot p_{\text{fix}}(1) - \frac{x}{n} \cdot p_{\text{fix}}(-1) \\ &= p_{\text{fix}}(1) \cdot \left(\frac{n-x}{n} - \frac{x}{n} \cdot \frac{p_{\text{fix}}(-1)}{p_{\text{fix}}(1)} \right). \end{aligned}$$

Using Lemma 4.3 we get

$$\Delta(x) = p_{\text{fix}}(1) \cdot \left(\frac{n-x}{n} - \frac{x}{n} \cdot e^{-2(N-1)\beta} \right)$$

and since $2(N-1)\beta \geq \ln(cn)$ with $c > 1$, we can bound $\Delta(x)$ from below by

$$\Delta(x) \geq p_{\text{fix}}(1) \cdot \left(\frac{n-x}{n} - \frac{1}{cn} \right).$$

To find the upper bound on the expected time that SSWM needs to find the fitness peak, we apply variable drift theorem (see Theorem 2.11) to the decreasing number of zeros $z = n - x$.

$$\Delta(x) \geq p_{\text{fix}}(1) \cdot \frac{zc - 1}{cn} = h(z)$$

The number of zeros changes from n (in the worst case scenario) to 1 (the last state that is not optimum), defining the boundaries of the integral

$$\begin{aligned}
E(T \mid X_0) &\leq \frac{1}{h(1)} + \int_1^n \frac{1}{h(z)} dz \\
&= \frac{1}{p_{\text{fix}}(1)} \cdot \frac{cn}{c-1} + \int_1^n \frac{1}{p_{\text{fix}}(1)} \cdot \frac{cn}{zc-1} dz \\
&= \frac{1}{p_{\text{fix}}(1)} \cdot \left(\frac{cn}{c-1} + n \cdot \ln\left(\frac{cn-1}{c-1} \right) \right) \\
&\leq \frac{1}{p_{\text{fix}}(1)} \cdot \left(O(n) + n \cdot \ln\left(n \cdot \frac{c}{c-1} \right) \right) \\
&= \frac{1}{p_{\text{fix}}(1)} \cdot \left(O(n) + n \cdot \ln(n) + n \cdot \ln\left(\frac{c}{c-1} \right) \right) \\
&= \frac{n \ln(n) + O(n)}{p_{\text{fix}}(1)}.
\end{aligned}$$

Alternatively, we can use p_{fix} bounds (see Lemma 4.2) to obtain

$$E(T \mid X_0) \leq \left(1 + \frac{1}{2\beta}\right) \cdot (n \ln(n) + O(n)). \qquad \square$$

The upper bound from Theorem 5.1 required $2(N-1)\beta \geq \ln(cn)$, or equivalently, $2N\beta \geq \ln(n) + \ln(c) + 2\beta$. This condition is vital since if $N\beta$ is chosen too small, the runtime of SSWM on OneMax is exponential with very high probability, as we show next.

If $2N\beta$ is smaller than $\ln(n)$ by a factor of $1 - \varepsilon$, for some constant $\varepsilon > 0$, the optimisation time is exponential in n, with overwhelming probability. SSWM therefore exhibits a phase transition behaviour: changing $N\beta$ by a constant factor makes a difference between polynomial and exponential expected optimisation times on OneMax (see Figure 5.1).

This result sets a limit to the complexity that can be evolved: for a fixed selection strength, there is a maximum number of sites that can be efficiently adapted. Typically, selection is deemed efficient when $N\Delta f > 1$ (corresponding to $N\beta$, in our framework). Our result defines the conditions for which selection is efficient in a multilocus setting, taking into account mutational pressure. It shows that, even if $N\beta > 1$ at every locus, for selection to be able to drive a population to the fitness peak $N\beta$ needs to scale non-linearly with the length of the target sequence ($2N\beta > \ln(cn)$).

Theorem 5.2. *If* $1 \le N\beta \le \frac{1-\varepsilon}{2} \ln n$ *for some* $0 < \varepsilon < 1$*, then the optimisation time of SSWM on* OneMax *is at least* $2^{cn^{\varepsilon/2}}$ *with probability* $1 - 2^{-\Omega(n^{\varepsilon/2})}$*, for some constant* $c > 0$.

The condition $N\beta \ge 1$ is used to ease the presentation; we believe it not to be essential and we believe it can be dropped when using more detailed calculations. The idea behind the proof of Theorem 5.2 is to show that for all search points with at least $n - n^{\varepsilon/2}$ ones, there is a negative drift for the number of ones. This is because for small $N\beta$ the selection pressure is too weak, and worsenings in fitness are more likely than steps where mutation leads the algorithm closer to the optimum. We then use the negative drift theorem with self-loops (Theorem 2.13 on page 38). Finally, note that the theorem uses "$p_{k,k\pm d} \le x$" as a shorthand for "$p_{k,k+d} \le x$ and $p_{k,k-d} \le x$".

Proof of Theorem 5.2. To prove this theorem, the negative drift theorem (Theorem 2.13) will be applied, taking the number of zeros as distance function to the optimum. Our notation refers to numbers of ones for simplicity. Let $p_{x,x\pm 1}$ be the probability that SSWM will make a transition from a search point with x ones to one with $x \pm 1$ ones, and assuming $x \ge n - n^{\varepsilon/2}$, then the expected drift towards the optimum is bounded as follows

$$p_{x,x+1} = \frac{n-x}{n} \cdot p_{\text{fix}}(1) \le n^{\varepsilon/2-1} \cdot p_{\text{fix}}(1)$$

since $p_{\text{fix}}(1) \le \frac{2\beta}{1-e^{-2N\beta}}$ (see Lemma 4.2)

$$p_{x,x+1} \le n^{\varepsilon/2-1} \cdot \frac{2\beta}{1-e^{-2N\beta}} \le n^{\varepsilon/2-1} \cdot \frac{2\beta}{1-e^{-2}}.$$

On the other hand,

$$\begin{aligned} p_{x,x-1} &\ge \frac{x}{n} \cdot p_{\text{fix}}(-1) \ge \frac{n-n^{\varepsilon/2}}{n} \cdot p_{\text{fix}}(-1) \\ &= p_{\text{fix}}(-1) \cdot \left(1 - n^{\varepsilon/2-1}\right) \end{aligned}$$

using $e^{2N\beta} \le e^{(1-\varepsilon)\ln n} = n^{1-\varepsilon}$

$$p_{x,x+1} \ge \frac{2\beta \cdot n^{\varepsilon}}{n} \cdot \left(1 - n^{\varepsilon/2-1}\right).$$

The expected drift $\Delta(x)$ is hence at most

$$\begin{aligned}\Delta(x) &\leq \frac{2\beta}{1-e^{-2}} \cdot n^{\varepsilon/2-1} - \frac{2\beta \cdot n^{\varepsilon}}{n} \cdot \left(1 - n^{\varepsilon/2-1}\right) \\ &= 2\beta \cdot n^{\varepsilon/2-1} \cdot \left(\frac{1}{1-e^{-2}} - n^{\varepsilon/2} \cdot \left(1 - n^{\varepsilon/2-1}\right)\right) \\ &= -\Omega(\beta \cdot n^{\varepsilon-1}).\end{aligned}$$

Now, the self-loop probability is at least $p_{x,x} = 1 - p_{x,x+1} - p_{x,x-1} = 1 - O(\beta n^{\varepsilon-1})$, hence the first condition of the negative drift theorem is satisfied. Since there are only local mutations the second condition on exponentially decreasing transition probabilities follows immediately. The negative drift theorem, applied to the number of zeros on an interval of $[0, n^{\varepsilon/2}]$, proves the claimed result. □

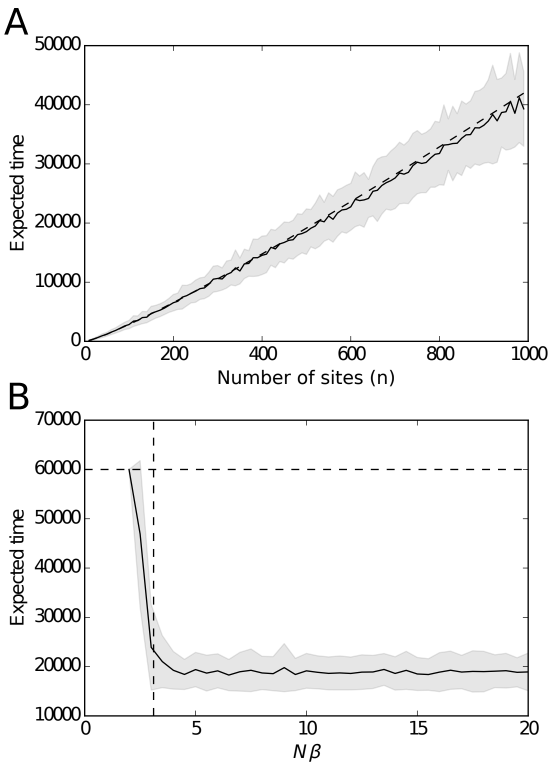


Fig. 5.1 (**A**) Expected time required to reach the fitness peak in the OneMax as a function of genome size. Solid black line represents the mean of 100 runs for given n and shaded area their standard deviation. Dashed line represents the theoretical upper bound on this expectation: $(1 + 1/(2\beta))n\ln(n) + n$. $N\beta$ was set to 100. (**B**) Black line represents the mean time to reach the fitness peak for a constant genome size ($n = 500$) and selection strength ($\beta = 0.1$), with increasing population size N, and shaded areas represent the standard deviation. Dashed line represents the critical value of selection strength ($2(N-1)\beta = \ln n$) separating the polynomial and exponential regimes for the time to reach the fitness peak. Simulations were stopped if they took longer than 6×10^4 iterations.

To conclude this section, we generalise the previous results to linear landscapes *regardless* of their distribution of mutational effects, for that, we introduce the family of linear functions. For elitist algorithms such as RLS or the (1+1) EA, linear functions are as easy as ONEMAX (Droste et al., 2002). This is due to the fact that their acceptance probability (see Equation (2.2)) only considers the sign of the fitness difference between the mutated and the parent genotype. However, for fitness dependent selection operators like those used by MA or SSWM (see Equations (2.3) and (4.2)), linear functions can be difficult as we will see in Theorem 5.3. Finally, note that ONEMAX is just the special case of a linear function when all the weights have value 1.

Definition 5.1 (LINEAR). *Let $x \in \{0,1\}^n$ be a bit-string and $w_{i=1,\dots,n} \in \mathbb{R}^+$ denote the contribution of each bit-position to the total fitness. Then,*

$$\textsc{Linear}(x) = \sum_{i=1}^{n} w_i x_i.$$

For ONEMAX we showed that $2(N-1)\beta \geq \ln(cn)$ for $c > 1$ is sufficient to get a positive drift. In the case of linear functions, for bits of weight at least w^* we get a positive drift on those bits if $2(N-1)\beta w^* \geq \ln(cn)$. Call all such bits *large effect sites* or *heavy*, then by the same arguments as for ONEMAX, SSWM optimises all heavy bits in the same time bound as for ONEMAX.

When $\beta = 1/\text{poly}(n)$, the only sites we cannot guarantee to fix in polynomial time are those with effect smaller than w^*, where w^* defines a threshold on the distribution of effects separating the loci "easily" adapted from the "small effect" ones. The total contribution of these sites is at most nw^*. Actually, when these site weights are small enough we have that $p_{\text{fix}}(+w) \approx p_{\text{fix}}(-w)$, and thus these bits will flip back and forth mimicking a random walk.

Theorem 5.3. *Let $w_1, \dots, w_n$ and $W := \sum_{i=1}^{n} w_i$. Then SSWM with $2(N-1)\beta w^* \geq \ln(cn)$ and $c > 1$ finds a solution of fitness at least*

$$\sum_{i=1}^{n-n^*} w_i \geq W - n^* w^* \geq W - nw^* = W\left(1 - w^* \frac{n}{W}\right)$$

in expected time at most

$$\frac{n\ln(n) + O(n)}{p_{\text{fix}}(w^*)} \leq \left(1 + \frac{1}{2\beta w^*}\right) \cdot (n\ln(n) + O(n))$$

where w^* is the minimum weight we want to optimise and n^* the number of weights with value less than w^*.

Proof. If $w^* \geq \frac{W}{n}$ the statement is trivial as then the lower bound on the fitness is non-positive.

Without loss of generality, we assume that the weights are ordered in ascending order: $w_1 \leq w_2 \leq \cdots \leq w_n$. Looking now when $w^* < \frac{W}{n}$ and ignoring the n^* weights such that $w_i < w^*$, $i = 1, \ldots, n^*$, (note that $w^* \in \left[\frac{\ln(cn)}{2(N-1)\beta}, \frac{W}{n}\right)$), we lower bound the positive drift by the probability of flipping one of the 0-bits with weight bigger than w^* times the fixation probability underestimated for the case where that bit has exactly weight of w^*

$$\Delta^+(x) \geq \frac{n - n^* - x}{n} \cdot p_{\text{fix}}(w^*).$$

For the backward drift, we look to the worst expected impact of one single bit $\frac{1}{n} \cdot p_{\text{fix}}(-w^*)$, then applying linearity of expectations we obtain

$$|\Delta^-(x)| \leq p_{\text{fix}}(-w^*).$$

The total expectation of the progress towards the optimum is therefore

$$\Delta(x) \geq p_{\text{fix}}(w^*) \left(\frac{n - n^* - x}{n} - \frac{p_{\text{fix}}(-w^*)}{p_{\text{fix}}(w^*)} \right).$$

Using Lemma 4.3 leads to

$$\Delta(x) \geq p_{\text{fix}}(w^*) \left(\frac{n - n^* - x}{n} - e^{-2(N-1)\beta w^*} \right)$$

and using $2(N-1)\beta w^* \geq \ln(cn)$, we get

$$\begin{aligned} \Delta(x) &\geq p_{\text{fix}}(w^*) \cdot \left(\frac{n - n^* - x}{n} - \frac{1}{cn} \right) \\ &= p_{\text{fix}}(w^*) \cdot \frac{c(n - n^* - x) - 1}{cn}. \end{aligned} \tag{5.1}$$

Now we apply variable drift to the number of zeros z in the $n - n^*$ bits that we want to optimise, i.e., $z = n - n^* - x$. We obtain the following drift bound which is always positive if $c > 1$.

$$\Delta(x) \geq p_{\text{fix}}(w^*) \cdot \frac{cz - 1}{cn} = h(z).$$

The integral range will go from the farthest point to the optimum (all of the $n-n^*$ heaviest weights being 0) to the closest (only 1 bit of the $n-n^*$ heaviest weights being 0)

$$\begin{aligned}
E(T \mid X_0) &\leq \frac{1}{h(1)} + \int_1^{n-n^*} \frac{1}{h(z)} dz \leq \frac{1}{h(1)} + \int_1^{n} \frac{1}{h(z)} dz \\
&= \frac{1}{p_{\text{fix}}(w^*)} \cdot \frac{cn}{c-1} + \int_1^{n} \frac{1}{p_{\text{fix}}(w^*)} \cdot \frac{cn}{cz-1} dz \\
&= \frac{1}{p_{\text{fix}}(w^*)} \cdot \left(\frac{cn}{c-1} + n \cdot \ln\left(\frac{cn-1}{c-1} \right) \right) \\
&\leq \frac{1}{p_{\text{fix}}(w^*)} \cdot \left(O(n) + n \cdot \ln\left(\frac{cn}{c-1} \right) \right) \\
&= \frac{1}{p_{\text{fix}}(w^*)} \cdot \left(O(n) + n \cdot \ln\left(\frac{c}{c-1} \right) + n\ln(n) \right) \\
&= \frac{n\ln(n) + O(n)}{p_{\text{fix}}(w^*)} \leq \left(1 + \frac{1}{2\beta w^*}\right) \cdot \frac{n\ln(n) + O(n)}{w^*},
\end{aligned}$$

where the last bound follows from Lemma 4.2. □

Since the $n-n^*$ sites of large effect behave essentially like the equal effects case, for a constant selection strength there is a maximum fitness that can be reached in $O(n \ln n)$. Reaching a fraction of this fitness takes linear time, whilst adapting further becomes highly inefficient as we show graphically with simulations (Figure 5.2). Without knowledge of the actual distribution of effects, it is impossible to determine n^* and, hence the fitness level that is guaranteed to be reached in polynomial time. However, since all effects are drawn from the same distribution, n^* will always be a constant fraction of n (since n^* is simply the fraction of weights below w^*, $n^* = \text{CDF}(w^*) \cdot n$). These scalings are valid for *any* distribution of effects and represent hard limits on this class of fitness functions.

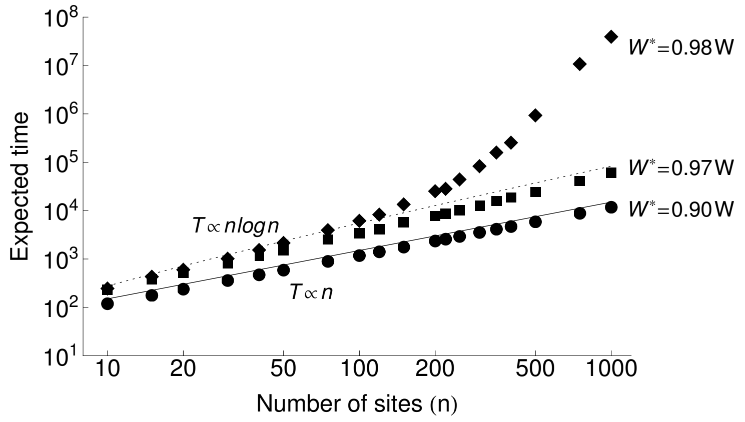


Fig. 5.2 Expected time to reach different fractions of the total fitness for an exponential distribution of effects. Data points correspond to means of 1000 runs, $N = 20$ and $\beta = 0.1$.

5.2 Fitness Ridges

One extreme form of epistatic landscape is when mutations need to be accumulated in a particular order, having no effect outside of this order (Kondrashov and Kondrashov, 2001). This creates a landscape in sequence space characterised by a fitness ridge and vast neutral plateaus leading to the optimum. A well-known example in EC with this feature is the LEADINGONES function, which counts the number of ones until the first appearance of a 0-bit (Rudolph, 1997a). We saw in Definition 2.5 that it can be mathematically expressed as

$$\textsc{LeadingOnes}(x) = \sum_{i=1}^{n} \prod_{j=1}^{i} x_j.$$

In order to increase its current fitness, it is necessary to flip the first zero in the genome to one. Flipping any other zero to one will result in a mutant offspring with the same fitness as its parent, while flipping any of the leading ones into zero can result in a drastic fitness loss. In this landscape, the probability of a beneficial mutation is $1/n$, as only flipping the first zero in the genome will result in a fitness increase. However, as more ones can follow this locus (neutral mutations that may have fixed neutrally), the increase in trait value can be higher than 1. On the other hand, mutating the j-th position of the x already well adapted sites will result in a fitness decrease of size $k = x - j + 1$ yielding a big negative impact on the drift. However as long as $N \geq 3$ the fixation probability decreases exponentially for deleterious mutations and can overcome the linear impact k of mutation (see Lemma 4.4).

The following theorem shows that even if the path to the optimum is narrow and mutations have to occur in a specific order, populations in the SSWM regime are able to climb the fitness peak relatively fast (polynomial time). Remarkably, this result holds for *any* selection strength above a constant value, indicating that, for landscapes of this type, there are no limits to the number of loci that can be adapted in polynomial time, as long as selection strength is above this constant value. The main reason for this is that, even though the number of deleterious mutations still increases as the population approaches the optimum, most of them are much less likely to be fixed due to their strong deleterious effects. This leads to a much less pronounced slowdown of the speed of adaptation as the population approaches the optimum. Notice that in this family of landscapes, the time to reach a fraction of the maximum fitness is also $O(n^2)$ (note that $p_{\text{fix}}(1) = \Omega(1)$ due to the required selection strength). Note that in order to obtain polynomial upper bounds we need the extra condition of $beta = 1/\text{poly}\,(n)$.

Theorem 5.4. *The expected optimisation time of SSWM with local mutations, $\beta \in \mathbb{R}^+$ and $2(N-1)\beta \geq 3$ on* LEADINGONES *is*

$$O\left(\frac{n^2}{p_{\text{fix}}(1)}\right) = O\left(n^2 \cdot \left(1 + \frac{1}{2\beta}\right)\right).$$

Proof. The forward and backward drift can be bounded as follows

$$\Delta^+(x) \geq \frac{1}{n} \cdot p_{\text{fix}}(1), \quad |\Delta^-(x)| = \frac{1}{n} \cdot \sum_{k=1}^{x} p_{\text{fix}}(-k).$$

To use Lemma 4.4 we need that $2\beta(N-1) \geq 1 + 2 \cdot \max\left(1, \frac{1}{\delta}\right)$. Since the minimum worsening δ in LEADINGONES is 1 the condition reduces to $2\beta(N-1) \geq 3$, thus

$$|\Delta^-(x)| \leq \frac{1}{n} \cdot \sum_{k=1}^{n} e^{-(k-1)} \cdot p_{\text{fix}}(-1) \leq \frac{1}{n} \cdot \frac{e}{e-1} \cdot p_{\text{fix}}(-1) \leq \frac{2}{n} \cdot p_{\text{fix}}(-1).$$

The total drift is

$$\Delta(x) \geq \frac{p_{\text{fix}}(1)}{n} \cdot \left(1 - 2 \cdot \frac{p_{\text{fix}}(-1)}{p_{\text{fix}}(1)}\right)$$

and using Lemma 4.3 leads to

$$\Delta(x) \geq \frac{p_{\text{fix}}(1)}{n} \cdot \left(1 - 2 \cdot e^{-2(N-1)\beta}\right).$$

Assuming $2(N-1)\beta \geq 3$ yields

$$\Delta(x) \geq \frac{p_{\text{fix}}(1)}{n} \cdot \left(1 - \frac{2}{e^3}\right) \geq \frac{9p_{\text{fix}}(1)}{10n}.$$

Now we apply variable drift theorem to the number of zeros z

$$\Delta(x) \geq \frac{9}{10n} \cdot p_{\text{fix}}(1) = h(z)$$
$$E(T \mid X_0) \leq \frac{10n}{9p_{\text{fix}}(1)} + \int_1^n \frac{10n}{9p_{\text{fix}}(1)} dz = O\left(\frac{n^2}{p_{\text{fix}}(1)}\right)$$

using p_{fix} bounds (see Lemma 4.2) one gets

$$E(T \mid X_0) \leq O\left(n^2 \cdot \left(1 + \frac{1}{2\beta}\right)\right).$$

□

5.3 Adaptation in a General Class of Landscapes

We now turn to a general class of fitness landscapes: unimodal functions. This class includes all functions that have only one maximum, meaning that it includes functions displaying arbitrary forms of epistasis, excluding only some types of sign epistasis which may lead to multiple peaks (Poelwijk et al., 2007; Weinreich et al., 2005), as mentioned before.

The defining feature of the members of the unimodal class is that any genotype other than the peak has at least one mutational neighbour (a genotype that differs exactly by one mutation) of higher fitness value. The following theorem proves an upper bound of $O(nd/p_{\text{fix}}(\delta))$, where d is the longest shortest path to the optimum and δ denotes the minimum of these trait increases (or decreases) in the landscape.

This bound depends on the length d, and as such is not independent of the instance of the function class we are considering. It should be noted that this bound can be loose, as can be seen by comparing to the previous results for linear functions (which are part of the unimodal function class): the fitness range d is of size n, entailing a bound for the time to adaptation of $O(n^2)$ when, in reality, the time on the linear function class grows slower $O(n \ln n)$ (see Theorem 5.3). On the other hand, it leads to a tight bound for the LeadingOnes function (see Theorem 5.4).

Moreover, the $O(nd/p_{\text{fix}}(\delta))$ upper bound does not guarantee that the time to reach the peak is polynomial: there could exist members of the unimodal function class for which d is exponential (Droste et al., 2006; Rudolph, 1997b), making the bound exponential. More generally, it is known that the black-box complexity of unimodal functions is exponential (Droste et al., 2006).

Theorem 5.5. *SSWM can optimise every unimodal function in*

$$O\left(\frac{nd}{p_{\text{fix}}(\delta)}\right) = O\left(nd \cdot \left(1 + \frac{1}{2\beta\delta}\right)\right)$$

where d is the longest shortest path to the optimum. Provided $2(N-1)\beta\delta \geq \ln(cn)$ with $c > 1$, $\beta \in R^+$ and $\delta \in R^+$ which is the minimum fitness improvement (and worsening).

Proof. Pessimistically assuming that only one specific bit flip leads to an improvement, the positive drift can be bounded by

$$\Delta^+(x) \geq \frac{1}{n} \cdot p_{\text{fix}}(\delta).$$

For the backward drift, we look to the worst expected impact of one single bit $\frac{1}{n} \cdot p_{\text{fix}}(-\delta)$, then applying linearity of expectations we obtain

$$|\Delta^{-}(x)| \leq \; p_{\text{fix}}(-\delta).$$

The total expectation of the progress towards the optimum is therefore

$$\Delta(x) \geq \; p_{\text{fix}}(\delta) \left(\frac{1}{n} - \frac{p_{\text{fix}}(-\delta)}{p_{\text{fix}}(\delta)} \right)$$

using Lemma 4.3 leads to

$$\Delta(x) \geq \; p_{\text{fix}}(\delta) \cdot \left(\frac{1}{n} - e^{-2(N-1)\beta\delta} \right)$$

and since $2(N-1)\beta\delta \geq \ln(cn)$ with $c > 1$

$$\begin{aligned} \Delta(x) &\geq \; p_{\text{fix}}(\delta) \cdot \left(\frac{1}{n} - \frac{1}{cn} \right) \\ &= \; \Omega \left(\frac{p_{\text{fix}}(\delta)}{n} \right). \end{aligned}$$

Now we apply variable drift theorem to the number of zeros z, integrating from the farthest point to the optimum ($z = d$) to the closest ($z = 1$)

$$\begin{aligned} \Delta(x) &= \Omega \left(\frac{p_{\text{fix}}(\delta)}{n} \right) = h(z), \\ E(T \mid X_0) &\leq O \left(\frac{n}{p_{\text{fix}}(\delta)} \right) + O \left(\int_1^d \frac{n}{p_{\text{fix}}(\delta)} dz \right) \\ &= O \left(\frac{nd}{p_{\text{fix}}(\delta)} \right) \end{aligned}$$

using p_{fix} bounds (see Lemma 4.2) we obtain an alternative formula

$$E(T \mid X_0) = O \left(nd \cdot \left(1 + \frac{1}{2\beta\delta} \right) \right).$$

□

5.4 Detecting the Steepest Slope

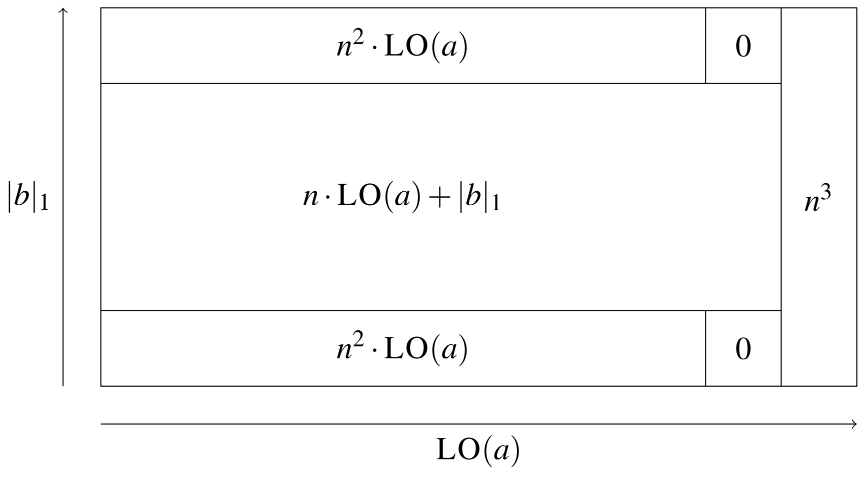
Finally, we investigate a feature that distinguishes SSWM from the (1+1) EA as well as the Metropolis algorithm: the fact that larger improvements are more likely to be accepted than smaller improvements. To this end, we consider the function BALANCE, originally introduced by Rohlfshagen et al. (2009) as an example where rapid dynamic changes in dynamic optimisation can be beneficial. The function has also been studied in the context of stochastic ageing by Oliveto and Sudholt (2014) and it goes back to an earlier idea by Witt (2008).

In its static (non-dynamic) form, BALANCE can be illustrated by a two-dimensional plane, whose coordinates are determined by the number of leading ones (LO) in the first half of the bit string, and the number of ones in the second half, respectively. The former has a steeper gradient than the latter, as the leading ones part is weighted by a factor of n in the fitness (see Figure 5.3).

Definition 5.2 (BALANCE Rohlfshagen et al., 2009)**.** *Let $a, b \in \{0,1\}^{n/2}$ and $x = ab \in \{0,1\}^n$. Then*

$$\textsc{Balance}(x) = \begin{cases} n^3 & \textit{if } \mathrm{LO}(a) = n/2, \textit{else} \\ |b|_1 + n \cdot \mathrm{LO}(a) & \textit{if } n/16 < |b|_1 < 7n/16, \textit{else} \\ n^2 \cdot \mathrm{LO}(a) & \textit{if } |a|_0 > \sqrt{n}, \textit{else} \\ 0 & \textit{otherwise} \end{cases}$$

where $|x|_1 = \sum_{i=1}^{n/2} x_i$, $|x|_0$ is a number of zeros and $\mathrm{LO}(x) := \sum_{i=1}^{n/2} \prod_{j=1}^{i} x_j$ counts the number of leading ones.



$n^2 \cdot \mathrm{LO}(a)$
0
$|b|_1$
$n \cdot \mathrm{LO}(a) + |b|_1$
n^3
$n^2 \cdot \mathrm{LO}(a)$
0
$\mathrm{LO}(a)$

Fig. 5.3 Visualisation of BALANCE Rohlfshagen et al. (2009).

The function is constructed in such a way that all points with a maximum number of leading ones are global optima, whereas increasing the number of ones in the second half beyond a threshold of $7n/16$ (or decreasing it below a symmetric threshold of $n/16$) leads to a trap, a region of local optima that is hard to escape from.

Rohlfshagen et al. (2009) showed the following lower bound for the (1+1) EA. The statement is specialised to non-dynamic optimisation and slightly strengthened by using a statement from their proof.

Theorem 5.6 (Theorem 3 in Rohlfshagen et al., 2009)**.** *With probability* $\Omega(1)$ *the (1+1) EA on* BALANCE *reaches a trap, and then needs at least* $n^{\sqrt{n}}$ *further generations in expectation to find an optimum from there. The expected optimisation time of the (1+1) EA is thus* $\Omega(n^{\sqrt{n}})$.

We believe that the probability bound $\Omega(1)$ can be strengthened to $1 - e^{-\Omega(n^{1/2})}$ with a more detailed analysis, which would show that the (1+1) EA gets trapped with an overwhelming probability.

We next show that SSWM with high probability finds an optimum in polynomial time. For appropriately small β we have sufficiently many successes on the LEADINGONES part such that we find an optimum before the ONEMAX-part reaches the region of local optima (see Figure 5.4). This is because for small β the probability of accepting small improvements is small. The fact that SSWM is slower than the (1+1) EA on ONEMAX (see Theorems 2.10 and 5.1) by a factor of $O(1/p_{\text{fix}}(1))$ turns into an advantage over the (1+1) EA on BALANCE.

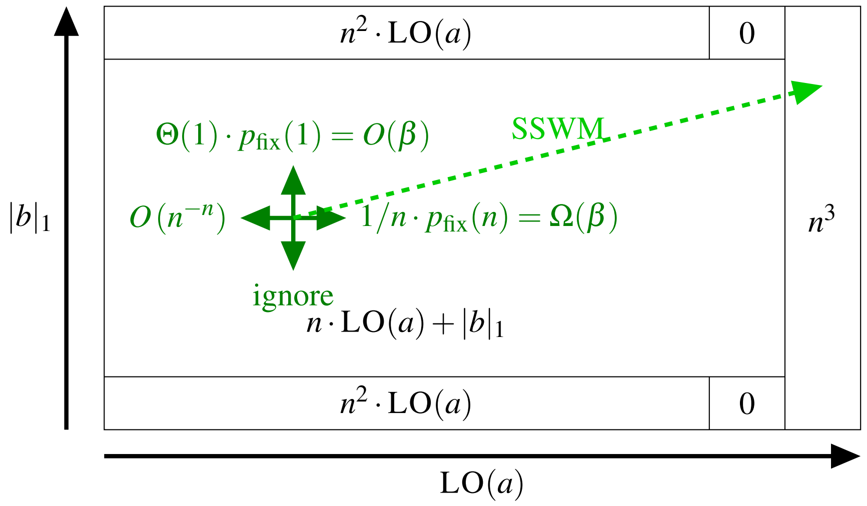

$n^2 \cdot \mathrm{LO}(a)$
0
$\Theta(1) \cdot p_{\text{fix}}(1) = O(\beta)$
SSWM
$|b|_1$
$O(n^{-n})$
$1/n \cdot p_{\text{fix}}(n) = \Omega(\beta)$
n^3
ignore
$n \cdot \mathrm{LO}(a) + |b|_1$
$n^2 \cdot \mathrm{LO}(a)$
0
$\mathrm{LO}(a)$

Fig. 5.4 Transition probabilities of the SSWM algorithm on BALANCE (see Lemmas 5.1 and 5.3).

The following lemma shows that SSWM effectively uses elitist selection on the LEADINGONES part of the function in a sense that every decrease is rejected with overwhelming probability. This scenario corresponds with the arrow pointing to the left on Figure 5.4.

Lemma 5.1. *For every $x = ab$ with $n/16 < |b|_1 < 7n/16$ and $\beta = n^{-3/2}$ and $N\beta = \ln n$, the probability of SSWM accepting a mutant $x' = a'b'$ with $\mathrm{LO}(a') < \mathrm{LO}(a)$ and $n/16 < |b'|_1 < 7n/16$ is $O(n^{-n})$.*

Proof. The loss in fitness is at least $n - (|b'|_1 - |b|_1) \geq n/2$. The probability of SSWM accepting such a loss is at most

$$p_{\text{fix}}(-n/2) \leq \frac{1 - e^{-2\beta(-n/2)}}{1 - e^{-2N\beta(-n/2)}} \leq \frac{e^{\beta n}}{e^{N\beta n} - 1}.$$

Assuming $\beta = n^{-3/2}$ and $N\beta = \ln n$, this is at most

$$\frac{e^{\frac{\sqrt{n}}{n}}}{n^n - 1} \leq \frac{e}{n^n - 1} = O(n^{-n}).$$

□

The following lemma establishes the optimisation time of the SSWM algorithm on either the ONEMAX or the LEADINGONES part of BALANCE.

Lemma 5.2. *Let $\beta = n^{-3/2}$ and $N\beta = \ln n$. With probability $1 - e^{-\Omega(n^{1/2})}$, SSWM optimises the* LEADINGONES *part or reaches the trap (all search points with fitness $n^2 \cdot \mathrm{LO}(a)$) within T steps.*

$$T := \frac{n^2}{4} \cdot \frac{1}{p_{\text{fix}}(n)} \cdot \left(1 + n^{-1/4}\right)$$

Proof. We use the method of typical runs (Wegener, 2003): we consider the typical behaviour of the algorithm, and show that events where the algorithm deviates from a typical run are very unlikely. A union bound over all such failure events proves the claimed probability bound.

With probability $1/n$ a local mutation will flip the first 0-bit. This increases the fitness by $k \cdot n$ where k is the number of consecutive 1-bits following this bit position after mutation. The latter bits are called *free riders* and it is well known (Lehre and Witt, 2012a, Lemma 1 and proof of Theorem 2) that the number of free riders follows a geometric distribution with parameter $1/2$, only capped by the number of bits to the end of the bit string a.

We assume that the number of leading ones is never decreased since the probability of accepting such a fitness decrease is $O(n^{-n})$ by Lemma 5.1. On the other hand, any increase by mutation is accepted with probability at least $p_{\text{fix}}(n)$. In a step, the probability of increasing the number of leading ones is hence at least $1/n \cdot p_{\text{fix}}(n)$ and the expected number of such improvements in

$$T := \frac{n^2}{4} \cdot \frac{1}{p_{\text{fix}}(n)} \cdot (1 + n^{-1/4})$$

steps is at least $n/4 + n^{3/4}/4$. By Chernoff bounds (see Lemma A.4 in Appendix A), the probability that less than $n/4+n^{3/4}/8$ improvements happen is $e^{-\Omega(n^{1/2})}$. Also the probability that during this number of improvements less than $n/4 - n^{3/4}/8$ free riders occur is $e^{-\Omega(n^{1/2})}$. If these two rare events do not happen, a LEADINGONES value of $n/2$ is reached before time T. Taking the union bound over all rare failure probabilities proves the claim. □

We now show that the ONEMAX part is not optimised before the LEADINGONES part. This yields an overall expected progress towards the optimum and not the traps as depicted in Figure 5.4.

Lemma 5.3. *Let $\beta = n^{-3/2}$, $N\beta = \ln n$, and T be as in Lemma 5.2. The probability that SSWM starting with a_0b_0 such that $n/4 \leq |b_0|_1 \leq n/4 + n^{3/4}$ creates a search point ab with $|b|_1 \leq n/16$ or $|b|_1 \geq 7n/16$ in T steps is $e^{-\Omega(n^{1/2})}$.*

The proof of Lemma 5.3 requires a careful and delicate analysis to show that the constant factors are small enough such that the stated thresholds for $|b|_1$ are not surpassed.

Proof of Lemma 5.3. We only prove that a search point with $|b|_1 \geq 7n/16$ is unlikely to be reached with the claimed probability. The probability for reaching a search point with $|b|_1 \leq n/16$ is clearly no larger, and a union bound for these two events leads to a factor of 2 absorbed in the asymptotic notation.

Note that using the bounds on p_{fix} from Lemma 4.2 and introducing $\beta = n^{-3/2}$ we have

$$p_{\text{fix}}(n) \geq \frac{2\beta n}{1+2\beta n} \geq \frac{1}{n^{1/2}}$$

for $n \geq 2$, hence

$$T \leq \frac{n^{3/2}}{4} \cdot \left(1 + n^{-1/4}\right) = O\left(n^{5/2}\right).$$

We call a step *improving* if the number of ones in b increases and the step is accepted. The probability p_1 that the ONEMAX value increases by 1 is at most

$$p_1 \leq p_{\text{fix}}(1) = \frac{1-e^{-2\beta}}{1-e^{-2N\beta}}$$

using $\beta = n^{-3/2}$ and $N\beta = \ln n$

$$\begin{aligned} p_1 &\leq \frac{n^{-3/2}}{1-n^{-2}} \\ &= \frac{1}{n^{3/2}-n^{-1/2}} \\ &= O\left(n^{-3/2}\right) \end{aligned}$$

Now, by Chernoff bounds, the probability of having more than $S := (1+n^{-1/4}) \cdot p_1 \cdot T$ improving steps in T steps is $e^{-\Omega(n^{1/2})}$. Using a Chernoff bound for geometric random variables (Doerr, 2011, Theorem 1.14), the probability of S improving steps yielding a total progress of at least $(1+n^{-1/4}) \cdot 4/3 \cdot S$ is $e^{-\Omega(n^{1/2})}$. If none of these rare events happen, the progress is at most

$$\begin{aligned} &(1+n^{-1/4}) \cdot \frac{4}{3} \cdot S \\ &(1+O(n^{-1/4})) \cdot \frac{4}{3} \cdot p_1 \cdot T \\ \leq\ &(1+O(n^{-1/4})) \cdot \frac{1.14}{9} \cdot n. \end{aligned}$$

which for large enough n is less than the distance $7n/16 - (n/4 + n^{3/4})$ to reach a point with $|b|_1 \geq 7n/16$ from initialisation. This proves the claim. □

Finally, we put the previous lemmas together into our main theorem that establishes that SSWM can optimise BALANCE in polynomial time.

Theorem 5.7. *With probability* $1 - e^{-\Omega(n^{1/2})}$ *SSWM with random initialisation,* $\beta = n^{-3/2}$ *and* $N\beta = \ln n$ *optimises* BALANCE *in time* $O\left(n^{5/2}\right)$.

Proof. By Chernoff bounds, the probability that for the initial solution $x_0 = a_0 b_0$ we have $n/4 - n^{3/4} \leq |b_0|_1 \leq n/4 + n^{3/4}$ is $1 - e^{-\Omega(n^{1/2})}$. We assume pessimistically that $n/4 \leq |b_0|_1 \leq n/4 + n^{3/4}$. Then Lemma 5.3 is in force, and with probability $1 - e^{-\Omega(n^{1/2})}$ within T steps, T as defined in Lemma 5.2, SSWM does not reach a trap or a search point with fitness 0. Lemma 5.2 then implies that with probability $1 - e^{-\Omega(n^{1/2})}$ an optimal solution with $n/2$ leading ones is found. □

Finally, to complete the picture we present experimental results for the (1+1) EA and SSWM on BALANCE. As shown in Theorems 5.6 and 5.7, SSWM can effectively optimise this problem while the (1+1) EA needs exponential time.

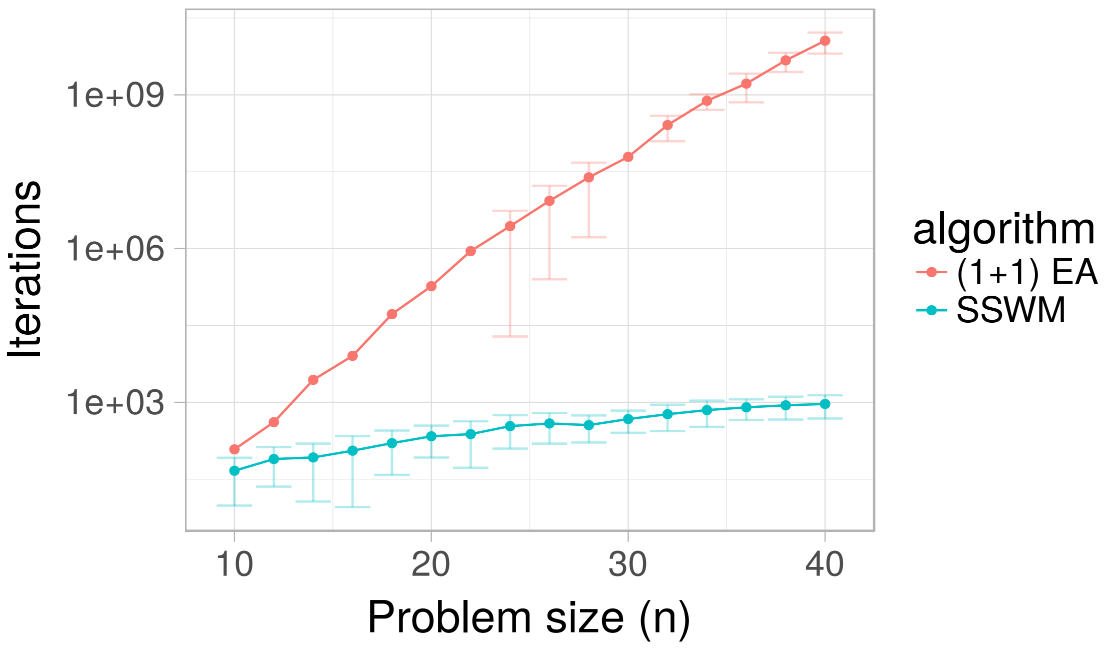


Fig. 5.5 Expected time required to reach an optimal search point of BALANCE for the (1+1) EA (red) and SSWM (blue). Results are averaged over 50 independent runs and error bars include ± one standard deviation. Simulations were stopped if they took longer than $n^{\sqrt{n}}$ iterations (recall Theorem 5.6) Parameters of SSWM were chosen in concordance with Theorem 5.7, i.e., $\beta = n^{-3/2}$ and $N\beta = \ln n$. A logarithmic scale with base 10 is used for the y-axis.

5.5 Conclusions

We have shown that, for a large class of fitness landscapes, it is *sufficient* that the selection strength $2N\beta$ is above the threshold $\ln n$ (along with $\beta = 1/\text{poly}\,(n)$) for populations to be able to efficiently climb to the fitness peak. We proved that, in the class of additive landscapes, this condition is both sufficient and necessary, implying a limit to the number of sites that can be efficiently adapted at a constant selection strength. Nevertheless, this critical threshold does not seem severe: selection strength should increase logarithmically with the number of sites under selection, indicating that a small increase in the selection gradient or population size translates in an exponential increase in the length of the sequences that can be evolved efficiently. Moreover, this condition is not always necessary: when considering a class of epistatic landscapes characterised by a single mutational path of strictly increasing fitness, we found that this limit no longer applies. A constant selection strength will enable a population to climb to the optimum, albeit at a slower rate than in an additive landscapes, regardless of the number of sites contributing to the trait. These results quantify the complexity of adaptive walks beyond linear landscapes or uncorrelated mutational neighbourhoods. They illustrate how the structure of the fitness landscape can impose limits to adaptation and how these stem directly from how the landscape conditions the distribution of effects of single mutants, in particular of deleterious mutations. Furthermore, they reveal how the buildup of mutational pressure that necessarily counteracts selection imposes a limit on the selection strength required for populations to overcome the entropic effects of mutation and make progress towards fitter genotypes.

From a computational perspective, we also showed that SSWM can take advantage of information about the steepest gradient outperforming classical evolutionary algorithms such as the (1+1) EA.

Chapter 6

When Non-Elitism Outperforms Elitism for Crossing Fitness Valleys

This chapter is mainly based on the following publications:

1. Oliveto, P. S., Paixão, T., Pérez Heredia, J., Sudholt, D., and Trubenová, B. (2016) When non-elitism outperforms elitism for crossing fitness valleys. *Proceedings of the Genetic and Evolutionary Computation Conference 2016 (GECCO '16)*, pages 1163–1170. ACM.

2. Oliveto, P. S., Paixão, T., Pérez Heredia, J., Sudholt, D., and Trubenová, B. (2017). How to Escape Local Optima in Black Box Optimisation: When Non-Elitism Outperforms Elitism. In *Algorithmica*. To appear.

As we studied in Chapter 2, evolutionary algorithms mainly differ in the way new solutions are generated (i.e. variation operators), how solutions are chosen for the next iterations (i.e. selection) and how many solutions are used by the heuristic in each iteration (i.e. population). Different variation operators, selection operators, population sizes and combinations of these lead to different algorithmic behaviours. In this chapter we analyse the effects of mutation and selection in overcoming local optima.

Two different approaches are commonly used by most black box algorithms. One strategy is to rely on variation operators such as mutation to produce new solutions of high fitness outside the basin of attraction of the local optimum. These are unary operators that construct a new candidate solution typically by flipping bits of an existing solution (see Subsection 2.2.1). Elitist algorithms (i.e. those that never discard the best found solution), mainly rely on such strategies when stuck on a local optimum. In a population-based algorithm different individuals may use different mutation rates to help escape local optima faster (Oliveto et al., 2009). Other variation operators may escape even faster than mutation. Population-based algorithms can recombine different solutions through the crossover operator to reach points outside the area of attraction of the local optima (Jansen and Wegener, 2002). This operation requires that sufficient diversity is available in the population which may be introduced by using some diversity-enforcing mechanism (Dang et al., 2017). Recently it has been shown that the interplay between the two variation operators, mutation and crossover, may efficiently give rise to the necessary burst of diversity without the need of any artificial diversity mechanism (Dang et al., 2017). Another combination that has been proven to be effective for elitist algorithms to overcome local optima is to alternate mutations with variable depth search (Sudholt, 2011a). A common approach used in practice is to restart the algorithm or perform several runs in parallel with the hope that the algorithm does not get stuck on the same local optima every time.

A very different approach is to attempt to escape by accepting solutions of lower fitness in the hope of eventually leaving the basin of attraction of the local optimum. This approach is the main driving force behind non-elitist algorithms. Compared to the amount of work on elitist black box algorithms, there are few theoretical works analysing the performance of non-elitism (see, e. g. Jansen, 2007; Neumann et al., 2009; Oliveto and Witt, 2014; Sasaki and Hajek, 1988). While both approaches may clearly be promising, it is still unclear when one should be preferred to the other.

In this chapter we investigate this topic by considering the areas between consecutive local optima, which we call *fitness valleys*. These valleys can have arbitrary length ℓ, i.e., , the distance between the local optima, and arbitrary depth d, i.e., the difference in function values between the optima and the point of minimum fitness between them. More precisely, we define a valley on a Hamming path (a path of Hamming neighbours) to ensure that mutation has the same probability of going forward on the path as going backwards. The valley is composed of a slope of length ℓ_1 descending towards a local minimum from which a slope of increasing fitness of length ℓ_2 can be taken to reach the end of the valley. The steepness of each slope is controlled by parameters d_1 and d_2, respectively indicating the fitness of the two local optima at the extreme left and extreme right of the valley. A sketch of a fitness valley is shown in Figure 6.1. Our aim is to analyse how the characteristics of the valley impact the performance of elitist versus non-elitist strategies.

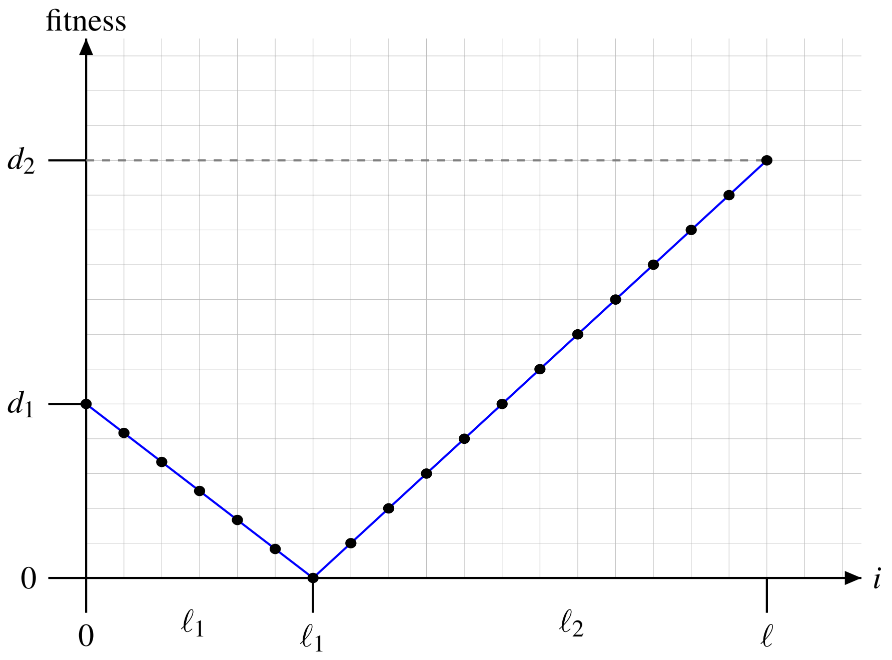


Fig. 6.1 Sketch of the function VALLEY.

We point out that understanding how to cross fitness valleys efficiently is a very important problem also in biology (Whitlock et al., 1995). Crossing fitness valleys represents one of the major obstacles to the evolution of complex traits. Many of these traits require accumulation of multiple mutations that are individually harmful for their bearers; a fitness advantage is achieved only when all mutations have been acquired—a fitness valley has been crossed.

The understanding of adaptation on highly rugged landscapes is still an open question in biology. The most famous attempt to shed light on this issue is Wright's shifting balance theory (SBT) (Wright, 1932). This theory suggests that isolated populations might fail to adapt on fitness landscapes with multiple peaks. However, by splitting into parallel populations, evolution could effectively cross the valleys and adapt on such landscapes. However, the SBT is still highly debated with supporting and dismissing literature (see e.g. (Goodnight and Wade, 2000) and (Coyne et al., 2000) respectively). Settling the debate on the SBT is not the purpose of this chapter since this thesis focused just on trajectory-based algorithms. However, we will identify fitness valleys' characteristics that represent intractable adaptive landscapes for isolated populations. We notice that this result is in concordance with the first claim of the SBT (isolated populations fail on highly rugged landscapes).

We consider the simple elitist (1+1) EA, the most-studied elitist evolutionary algorithm, and compare its ability to cross fitness valleys with the Strong Selection Weak Mutation regime (see Subsection 3.2.3) via a runtime analysis of the SSWM algorithm (see Algorithm 4.1). We show that greater speed-ups can be achieved by SSWM on fitness valleys.

After introducing the fitness functions and their encoding, we build upon Gambler's Ruin theory (see Subsection 2.3.2) to devise a general mathematical framework for the analysis of non-elitist algorithms using local mutations for crossing fitness valleys. We use it to rigorously show that SSWM is able to efficiently perform a random walk across the valley using only local mutations by accepting worse solutions, provided that the valley is not too deep. On the other hand, the (1+1) EA cannot accept worse solutions and therefore relies on global mutations to reach the other side of the valley in a single jump. More precisely, the (1+1) EA needs to make a jump across all valley points that have lower fitness; we call this the *effective length* of the valley.

As a result, the runtime of the (1+1) EA is exponential in the effective length of the valley while the runtime of SSWM depends crucially on the depth of the valley. We demonstrate the generality of the presented mathematical tool by using it to prove that the same asymptotic results achieved by SSWM also hold for the well-known Metropolis algorithm (MA) (Algorithm 2.7) that, differently from SSWM, always accepts improving moves. In Chapter 7, we will further investigate the similarities and differences of the SSWM and Metropolis algorithms. Jansen and Wegener (2007) previously compared the performance of the (1+1) EA and MA for a fitness valley encoded as a unitation function where the slopes are symmetric and of the same length. They used their fitness valley as an example where the performance of the two algorithms is asymptotically equivalent.

The framework also allows the analysis for concatenated "paths" of several consecutive valleys, creating a rugged fitness landscape that loosely resembles a "big valley" structure found in many problems from combinatorial optimisation (Boese et al., 1994; Merz and Freisleben, 1998; Ochoa and Veerapen, 2016; Reeves, 1999). In particular, in Section 6.3 we use it to prove that SSWM and MA can cross consecutive paths in expected time that depends crucially on the depth and number of the valleys.

Finally, we allow SSWM to use global mutations and show how it effectively combines non-elitism with global mutations to overcome sharp valleys. We will use the problem CLIFF_d, a function defined such that non-elitist algorithms have a chance to jump down a "cliff" of height roughly d and to traverse a fitness valley of Hamming distance d to the optimum. The function is a generalised construction of the unitation function (a function that only depends on the number of 1-bits in the bit string) introduced by Jägersküpper and Storch (2007) to give an example class of functions where a (1,λ) EA outperforms a (1+λ) EA. This analysis revealed that SSWM can cross the fitness valley. However, upon comparison with the (1+1) EA, SSWM achieved only a small speed-up: the expected time (number of function evaluations) of SSWM is at most $n^d/e^{\Omega(d)}$, while the (1+1) EA requires $\Theta(n^d)$.

6.1 Long Paths

Previous work on valley crossing (see e.g. Jägersküpper and Storch, 2007; Jansen, 2007) used functions of unitation to encode fitness valleys, with 1^n being a global optimum. The drawback of this construction is that the transition probabilities for mutation heavily depend on the current position. The closer an algorithm gets to 1^n, the larger the probability of mutation decreasing the number of ones and moving away from the optimum.

We follow a different approach to avoid this mutational bias, and to ensure that the structure of the fitness valley is independent of its position in the search space. This also allows us to easily concatenate multiple valleys.

We base our construction on the so-called *long k-paths*, paths of Hamming neighbours with increasing fitness whose length can be exponential in n. These paths were introduced and investigated experimentally by Horn et al. (1994) and subsequently formalised and rigorously analysed by Rudolph (1997b). Exponential lower bounds were shown by Droste et al. (2002). An example of a long k-path is shown in Table 6.1. The following formal, slightly revised definition is taken from (Sudholt, 2009, page 2517).

Definition 6.1. *Let $k \in \mathbb{N}$ and n be a multiple of k. The long k-path of dimension n is a sequence of bit strings from $\{0,1\}^n$ defined recursively as follows. The long k-path of dimension 0 is the empty bit string. Assume the long k-path of dimension $n-k$ is given by the sequence $\mathscr{P}^k_{n-k} = (p_1, \dots, p_\ell)$, where $p_1, \dots, p_\ell \in \{0,1\}^{n-k}$ and ℓ is the length of $\mathscr{P}^k_{n-k}$. Then the long k-path of dimension n is defined by prepending k bits to these search points: let $S_0 := (0^k p_1, 0^k p_2, \dots, 0^k p_\ell)$, $S_1 := (1^k p_\ell, 1^k p_{\ell-1}, \dots, 1^k p_1)$, and $B := (0^{k-1}1p_\ell, 0^{k-2}1^2 p_\ell, \dots, 01^{k-1}p_\ell)$. The search points in S_0 and S_1 differ in the k leading bits and the search points in B represent a bridge between them. The long k-path of dimension n, $\mathscr{P}^k_n$, is the concatenation of S_0, B, and S_1.*

The length of $\mathscr{P}^k_n$ is $k \cdot 2^{n/k} - k + 1$ (Sudholt, 2009, Lemma 3), which is exponential in n if, for example, $k = \Theta(\sqrt{n})$. An exponential length implies that the path has to be folded in $\{0,1\}^n$ in a sense that there are $i < j$ such that the i-th and the j-th point on the path have Hamming distance $\mathrm{H}(\cdot,\cdot)$ smaller than $j-i$. Standard bit mutations have a positive probability of jumping from the i-th to the j-th point, hence there is a chance to skip large parts of the path by taking a shortcut. However, long k-paths are constructed in such a way that at least k bits have to flip simultaneously in order to take a shortcut of length at least k. The probability of such an event is exponentially small if $k = \Theta(\sqrt{n})$, in which case the path still has exponential length.

Long k-paths turn out to be very useful for our purposes. If we consider the first points of a long k-path and assign increasing fitness values to them, we obtain a fitness-increasing path of any desired length (up to exponential in n (Sudholt, 2009, Lemma 3)).

P_0: 000000000	P_6 : 000111111	P_{12}: 111111000	P_{18}: 111000111
P_1: 000000001	P_7 : 000111011	P_{13}: 111111001	P_{19}: 111000011
P_2: 000000011	P_8 : 000111001	P_{14}: 111111011	P_{20}: 111000001
P_3: 000000111	P_9 : 000111000	P_{15}: 111111111	P_{21}: 111000000
P_4: 000001111	P_{10}: 001111000	P_{16}: 111011111	
P_5: 000011111	P_{11}: 011111000	P_{17}: 111001111	

Table 6.1 Example of a long k-path for $n = 9$ and $k = 3$: $\mathscr{P}_9^3 = (P_0, P_1, \dots, P_{21})$.

Given two points $\mathrm{P}_s, \mathrm{P}_{s+i}$ for $i > 0$, P_{s+i} is called the i-th *successor* of P_s and P_s is called a *predecessor* of P_{s+i}. Long k-paths have the following properties.

Lemma 6.1 (Long Paths)**.**

1. *For every $i \in \mathbb{N}_0$ and path points P_s and P_{s+i}, if $i < k$ then $H(\mathrm{P}_s, \mathrm{P}_{s+i}) = i$, otherwise $H(\mathrm{P}_s, \mathrm{P}_{s+i}) \geq k$.*
2. *The probability of a standard bit mutation turning P_s into P_{s+i} (or P_{s+i} into P_s) is $1/n^i \cdot (1 - 1/n)^{n-i}$ for $0 \leq i < k$ and the probability of reaching any search point in $\{\mathrm{P}_{s+i} \mid i \geq k\}$ from P_s is at most $1/(k!)$.*

Proof. The first statement was shown in (Sudholt, 2009, page 2517) (refining a previous analysis in Droste et al., 2002, page 73). The second statement follows from the first one, using that the probability of mutating at least k bits is at most $\binom{n}{k} n^{-k} \leq 1/(k!)$. □

In the following, we fix $k := \sqrt{n}$ such that the probability of taking a shortcut on the path is exponentially small. We assign fitness values such that all points on the path have a higher fitness than those off the path. This fitness difference is made large enough such that the considered algorithms are very unlikely to ever fall off the path. Assuming that we want to use the first m path points $\mathrm{P}_0, \dots, \mathrm{P}_{m-1}$, then the fitness is given by

$$f(x) := \begin{cases} h(i) & \text{if } x = \mathrm{P}_i, i < m \\ -\infty & \text{otherwise} \end{cases}$$

where $h(i)$ gives the fitness (height) of the i-th path point.

Then, assuming the algorithm is currently on the path, the fitness landscape is a one-dimensional landscape where (except for the two ends) each point has a Hamming neighbour as predecessor and a Hamming neighbour as successor on the path. Local mutations will create each of these with equal probability $1/n$. If we call these steps *relevant* and ignore all other steps, we get a stochastic process where in each relevant step we create a mutant up or down the path with probability $1/2$ each (for the two ends we assume a self-loop probability of $1/2$). The probability whether such a move is accepted then depends on the fitness difference between these path points.

It then suffices to study the expected number of relevant steps, as we obtain the expected number of function evaluations by multiplying with the expected waiting time $n/2$ for a relevant step.

Lemma 6.2. *Let $E[T]$ be the expected number of relevant steps for any algorithm described by Algorithm 2.3 with local mutations finding a global optimum. Then the respective expected number of function evaluations is $n/2 \cdot E[T]$, unless the algorithm falls off the path.*

In the following, we assume that all algorithms start on P_0. This behaviour can be simulated from random initialisation with high probability by embedding the path into a larger search space and giving hints to find the start of the path within this larger space (Sudholt, 2009). As such a construction is cumbersome and does not lead to additional insights, we simply assume that all algorithms start in P_0.

6.2 Crossing Simple Valleys

As mentioned earlier we will use the VALLEY problem (Figure 6.1). On the first slope starting at point P_0 the fitness decreases from the initial height $d_1 \in \mathbb{R}^+$ until the path point P_{ℓ_1} with fitness 0. Then the second slope begins with fitness increasing up to the path point $\mathrm{P}_{\ell_1+\ell_2}$ of fitness $d_2 \in \mathbb{R}^+$. The total length of the path is $\ell = \ell_1 + \ell_2$. We call such a path VALLEY.

Definition 6.2 (VALLEY). *Let $d_1, d_2 \in \mathbb{R}^+$ and $\ell_1, \ell_2 \in \mathbb{N}$, then the fitness $h(i)_{\text{VALLEY}}$ of the i-th path point is given by*

$$h(i)_{\text{VALLEY}} := \begin{cases} d_1 - i \cdot \frac{d_1}{\ell_1} & \text{if } i \leq \ell_1 \\ (i - \ell_1) \cdot \frac{d_2}{\ell_2} & \text{if } \ell_1 < i \leq \ell. \end{cases}$$

Here, $\frac{d_1}{\ell_1}$ and $\frac{d_2}{\ell_2}$ indicate the steepness of the two slopes (see Figure 6.1). We will use $d_2 > d_1$ to force the point P_ℓ to be the optimum.

6.2.1 Analysis for the (1+1) EA

We first show that the runtime of the (1+1) EA depends on the effective length ℓ^* of the valley, defined as the distance between the initial point P_0 and the first valley point of greater or equal fitness. Here we restrict parameters to $\ell_1 + \ell_2 \leq \sqrt{n}/4$, as then the probability of the (1+1) EA taking a shortcut is no larger than the probability of jumping by a distance of $\ell_1 + \ell_2$: $\frac{1}{(\sqrt{n})!} \leq n^{-\sqrt{n}/4}$ for $n \geq 4$.

Theorem 6.1. *Assume $\ell_1 + \ell_2 \leq \sqrt{n}/4$ and $d_1 \leq d_2$. The expected time for the (1+1) EA starting in P_0 to cross the fitness valley is $\Theta(n^{\ell^*})$ where $\ell^* = \ell_1 + \lceil d_1 \ell_2 / d_2 \rceil$.*

Proof. Let us first recall that due to its elitism the (1+1) EA can not fall off the path. To cross the fitness valley the (1+1) EA needs to jump from P_0 to a point with higher or equal fitness, thus it has to jump at least a distance ℓ^*. The probability of such a jump can be bounded from below using Lemma 6.1 by

$$\begin{aligned} p_{\text{jump}} &\geq n^{-\ell^*} \left(1 - \frac{1}{n}\right)^{n-\ell^*} \\ &\geq e^{-1} n^{-\ell^*} \end{aligned} \tag{6.1}$$

resulting in an expected time needed to jump over the valley of at most $en^{\ell^*} = O(n^{\ell^*})$. After jumping over the valley, the (1+1) EA has to climb at most the remaining $\left(1 - \frac{d_1}{d_2}\right)\ell_2 \leq \ell_2$ steps, and each improvement has a probability of at least $1/(en)$. The expected time for this climb is thus at most $e\ell_2 n$. As $\ell_2 < n$ and $\ell^* \geq \ell_1 \geq 2$, this time is $O(n^{\ell^*})$.

Note that, in case P_{ℓ^*} has the same fitness as P_0, the (1+1) EA can jump back to the beginning of the path, in which case it needs to repeat the jump. However, conditional on leaving P_{ℓ^*}, the probability that a successor is found is at least $\Omega(1)$. Hence in expectation $O(1)$ jumps are sufficient.

Furthermore, the probability of the jump can be bounded from above by the probability of jumping to any of the next potential $\sqrt{n}$ path points and by the probability of taking a shortcut (see Lemma 6.1)

$$\begin{aligned} p_{\text{jump}} &\leq \frac{1}{\sqrt{n}!} + \sum_{i=\ell^*}^{\sqrt{n}} n^{-i} \left(1 - \frac{1}{n}\right)^{n-i} \\ &= O(n^{-\ell^*}) \end{aligned}$$

where we used $\frac{1}{(\sqrt{n})!} \leq n^{-\sqrt{n}/4} \leq n^{-\ell^*}$. Thus the expected time is $\Omega(n^{\ell^*})$. □

6.2.2 A General Framework for Local Search Algorithms

We introduce a general framework to analyse the expected number of relevant steps of non-elitist local search algorithms (Algorithm 2.3 with local mutations) for the VALLEY problem. As explained in Section 6.1, in a relevant step mutation creates a mutant up or down the path with probability $1/2$, and this move is accepted with a probability that depends only on the fitness difference. For slopes where the gradient is the same at every position, this resembles a gambler's ruin process.

To apply classical gambler ruin theory (see Theorem 2.3) two technicalities need to be taken into account. Firstly, two different gambler ruin games need to be considered, one for descending down the first slope and another one for climbing up the second slope. The process may alternate between these two ruin games as the extreme ends of each game at the bottom of the valley are not absorbing states. Secondly, a non-elitist algorithm could reject the offspring individual even when it has a higher fitness than its parent. Hence the probabilities of winning or losing a dollar (i.e., the probabilities of moving one step up or down in the slope) do not necessarily add up to one. Here, loop probabilities of neither winning or losing a dollar need to be taken into account when estimating expected times (winning probabilities are unaffected by self-loops).

Theorem 6.2 (Gambler's Ruin with self-loops). *Consider a game where two players start with $n_1 \in \mathbb{N}^+$ and $n_2 \in \mathbb{N}^+$ dollars respectively. In each iteration player 1 wins one of player's 2 dollars with probability p_1, player 2 wins one of player's 1 dollars with probability p_2, and nothing happens with probability $1 - p_1 - p_2$. Then the probability of player 1 winning all the dollars before going bankrupt is:*

$$P_1 = \begin{cases} \frac{n_1}{n_1+n_2} & \text{if } p_1 = p_2 \\ \frac{1-\left(\frac{p_2}{p_1}\right)^{n_1}}{1-\left(\frac{p_2}{p_1}\right)^{n_1+n_2}} & \text{if } p_1 \neq p_2. \end{cases}$$

The expected time until either of both players become bankrupt i.e. the expected duration of the game is

$$E[T] = \begin{cases} \frac{n_1 n_2}{p_1+p_2} & \text{if } p_1 = p_2 \\ \frac{n_1-(n_1+n_2)P_1}{p_2^2-p_1^2} & \text{if } p_1 \neq p_2. \end{cases}$$

Proof. The proof follows directly from the results of the standard problem ($p_1 + p_2 = 1$), see Theorem 2.3. The only effect of the self-loops is to add extra iterations in the problem where nothing happens, therefore the winning probabilities will not be affected, however the expected duration of the game will be increased by the waiting time needed for a relevant iteration $1/(p_1+p_2)$. □

In order to simplify the calculations and improve the readability of this chapter we have developed the following notation.

Definition 6.3 (Framework's notation)**.** *The* VALLEY *problem can be considered as a Markov chain with states* $\{P_0, P_1, \dots, P_{\ell_1-1}, P_{\ell_1}, P_{\ell_1+1}, \dots, P_{\ell_1+\ell_2}\}$*. For simplicity we will sometimes refer to these points only with their sub-indices* $\{0,1,\dots,\ell_1-1,\ell_1,\ell_1+1,\dots,\ell_1+\ell_2\}$*.*

For any stochastic process on the VALLEY *problem we will denote by:*

(1) $p_{i\to j}$ *the probability of moving from state i to* $j \in \{i-1, i, i+1\}$ *in one iteration,*

(2) $p^{\text{GR}}_{i\to k}$ *the probability of a Gambler's Ruin process starting in i finishing in k before reaching the state* $i-1$*,*

(3) $E\left[T^{\text{GR}}_{i,k}\right]$ *the expected duration until either the state* $i-1$ *or k is reached,*

(4) $E[T_{i\to m}]$ *the expected time to move from state i to state m.*

We now present some lemmas that will simplify the runtime analysis of any trajectory-based optimiser (Algorithm 2.3) with local mutations, as long as some reasonable conditions on the selection operator are met. After stating the first lemma we will discuss in detail these conditions.

Lemma 6.3. *Consider any algorithm described by Algorithm 2.3 with local mutations and the following properties on* VALLEY *with* $\ell_1, \ell_2 \in \{2,3,\dots\}$ *and* $d_1, d_2 \in R^+$*.*

(i) $p_{\ell_1\to\ell_1-1},\ p_{\ell_1\to\ell_1+1} = \Omega(1)$

(ii) $p_{\ell_1\to\ell_1+1} \geq p_{\ell_1+1\to\ell_1} + \varepsilon$*, for* $\varepsilon > 0$ *a constant*

(iii) $p_{\text{acc}}(\Delta f)$ *is non-decreasing.*

Then the expected number of relevant steps for such a process to reach the point $P_{\ell_1+\ell_2}$ *starting from* P_0 *is*

$$E[T_{0\to\ell_1+\ell_2}] = \Theta(E[T_{1\to\ell_1}]) + \Theta(\ell_2).$$

Property (iii) describes a common feature of optimisation algorithms: the selection operator prefers fitness increases over decreases (e.g. Randomised Local Search, (1+1) EA or MA). Then, the bottleneck of VALLEY seems to be climbing down the first ℓ_1 steps since several fitness decreasing mutations have to be accepted.

Once at the bottom of the valley P_{ℓ_1} the process must keep moving. It could be the case that the algorithm climbs up again to P_0. But under some mild conditions it will only have to repeat the experiment a constant number of times (property (i) of the following lemma).

Finally, the algorithm will have to climb up to $P_{\ell_1+\ell_2}$. This will take linear time in ℓ_2, provided the probability of accepting an improvement $p_{\ell_1 \to \ell_1+1}$ is by a constant greater than accepting a worsening of the same size $p_{\ell_1+1 \to \ell_1}$, as required by property (ii).

Consider an algorithm with a selection operator that satisfies condition (iii) such as MA or SSWM. In order to satisfy the first two conditions, the selection strength must be big enough to accept the two possible fitness increases of VALLEY (d_1/ℓ_1 and d_2/ℓ_2) with constant probability. As we will see at the end of this section, this condition directly translates to $\beta d_1/\ell_1$, $\beta d_2/\ell_2 = \Omega(1)$ for SSWM and $\alpha d_1/\ell_1$, $\alpha d_2/\ell_2 = \Omega(1)$ for MA.

In order to prove the previous lemma we will make use of the following lemma that shows some implications of the conditions from the previous lemma.

Lemma 6.4. *In the context of Lemma 6.3, properties (i) and (ii) imply that*

(i) $p_{\ell_1 \to \ell_1-1} + p_{\ell_1 \to \ell_1+1} = 1/c_1$ *for some constant* $c_1 \geq 1$

(ii) $1 - c_1 \cdot p^{\mathrm{GR}}_{\ell_1+1 \to \ell_1} = 1/c_2$ *for some constant* $c_2 > 1$

(iii) $1 - c_1 c_2 \cdot p_{\ell_1 \to \ell_1-1} = 1/c_3$ *for some constant* $c_3 > 1$.

We believe that the proof of Lemma 6.4 is mainly technical and does not shed much light on the understanding on the problem. Hence, for the sake of readability we do not include it here, however it can be found in the Appendix 6.A.

Proof of Lemma 6.3. Since the algorithm only produces points in the Hamming neighbourhood it will have to pass through all the states on the path. We break down the set of states in three sets and expand the total time as the sum of the optimisation time for those three sets:

$$\mathrm{E}\left[T_{0 \to \ell_1+\ell_2}\right] = \mathrm{E}\left[T_{0 \to 1}\right] + \mathrm{E}\left[T_{1 \to \ell_1}\right] + \mathrm{E}\left[T_{\ell_1 \to \ell_1+\ell_2}\right]. \tag{6.2}$$

Note that the lower bound follows directly. Let us now consider the upper bound. We start using a recurrence relation for the last term: once in state ℓ_1, after one iteration, the algorithm can either move to state ℓ_1+1 with probability $p_{\ell_1 \to \ell_1+1}$, move to state $\ell_1 - 1$ with probability $p_{\ell_1 \to \ell_1-1}$ or stay in state ℓ_1 with the remaining probability (if the mutation is not accepted).

$$\begin{aligned} &\mathrm{E}\left[T_{\ell_1 \to \ell_1+\ell_2}\right] = \\ &1 + p_{\ell_1 \to \ell_1+1} \cdot \mathrm{E}\left[T_{\ell_1+1 \to \ell_1+\ell_2}\right] + p_{\ell_1 \to \ell_1-1} \cdot \mathrm{E}\left[T_{\ell_1-1 \to \ell_1+\ell_2}\right] + p_{\ell_1 \to \ell_1} \cdot \mathrm{E}\left[T_{\ell_1 \to \ell_1+\ell_2}\right]. \end{aligned}$$

Using $\mathrm{E}[T_{\ell_1-1\to\ell_1+\ell_2}] \leq \mathrm{E}[T_{0\to\ell_1+\ell_2}]$ this expression reduces to

$$\leq 1 + p_{\ell_1\to\ell_1+1}\cdot\mathrm{E}[T_{\ell_1+1\to\ell_1+\ell_2}] + p_{\ell_1\to\ell_1-1}\cdot\mathrm{E}[T_{0\to\ell_1+\ell_2}] + p_{\ell_1\to\ell_1}\cdot\mathrm{E}[T_{\ell_1\to\ell_1+\ell_2}].$$

Solving the previous expression for $\mathrm{E}[T_{\ell_1\to\ell_1+\ell_2}]$ leads to

$$\mathrm{E}[T_{\ell_1\to\ell_1+\ell_2}] \leq \frac{1 + p_{\ell_1\to\ell_1+1}\cdot\mathrm{E}[T_{\ell_1+1\to\ell_1+\ell_2}] + p_{\ell_1\to\ell_1-1}\cdot\mathrm{E}[T_{0\to\ell_1+\ell_2}]}{p_{\ell_1\to\ell_1-1} + p_{\ell_1\to\ell_1+1}}.$$

Since property (i) of Lemma 6.3 implies that the denominator is a constant $1/c_1$, we get

$$\mathrm{E}[T_{\ell_1\to\ell_1+\ell_2}] \leq c_1\left(1 + p_{\ell_1\to\ell_1+1}\cdot\mathrm{E}[T_{\ell_1+1\to\ell_1+\ell_2}] + p_{\ell_1\to\ell_1-1}\cdot\mathrm{E}[T_{0\to\ell_1+\ell_2}]\right). \tag{6.3}$$

Let us now focus on the term $\mathrm{E}[T_{\ell_1+1\to\ell_1+\ell_2}]$. Since the acceptance probability is a function of Δf, for both sides of the valley the probabilities of moving to the next or previous state remain constant during each slope and we can cast the behaviour as a Gambler's Ruin problem. Then, when the state is P_{ℓ_1+1} a Gambler's Ruin game (with self-loops) occurs. The two possible outcomes are: (1) the problem is optimised or (2) we are back in P_{ℓ_1}. Hence,

$$\mathrm{E}[T_{\ell_1+1\to\ell_1+\ell_2}] = \mathrm{E}\left[T^{\mathrm{GR}}_{\ell_1+1,\ell_1+\ell_2}\right] + p^{\mathrm{GR}}_{\ell_1+1\to\ell_1}\cdot\mathrm{E}[T_{\ell_1\to\ell_1+\ell_2}]. \tag{6.4}$$

Now we introduce (6.4) in (6.3), obtaining

$$\begin{aligned}\mathrm{E}[T_{\ell_1\to\ell_1+\ell_2}] \leq{}& c_1\left(1 + p_{\ell_1\to\ell_1-1}\cdot\mathrm{E}[T_{0\to\ell_1+\ell_2}]\right)\\ &+ c_1\cdot p_{\ell_1\to\ell_1+1}\cdot\left(\mathrm{E}\left[T^{\mathrm{GR}}_{\ell_1+1,\ell_1+\ell_2}\right] + p^{\mathrm{GR}}_{\ell_1+1\to\ell_1}\cdot\mathrm{E}[T_{\ell_1\to\ell_1+\ell_2}]\right).\end{aligned}$$

Solving for $\mathrm{E}[T_{\ell_1\to\ell_1+\ell_2}]$ yields

$$\mathrm{E}[T_{\ell_1\to\ell_1+\ell_2}] \leq \frac{c_1\left(1 + p_{\ell_1\to\ell_1+1}\cdot\mathrm{E}\left[T^{\mathrm{GR}}_{\ell_1+1,\ell_1+\ell_2}\right] + p_{\ell_1\to\ell_1-1}\cdot\mathrm{E}[T_{0\to\ell_1+\ell_2}]\right)}{1 - c_1\cdot p_{\ell_1\to\ell_1+1}\cdot p^{\mathrm{GR}}_{\ell_1+1\to\ell_1}}.$$

By Lemma 6.4, properties (i) and (ii) of Lemma 6.3 imply that the denominator is a constant $1/c_2$. Hence,

$$\begin{aligned}\mathrm{E}[T_{\ell_1\to\ell_1+\ell_2}] &\leq c_1c_2\cdot\left(1 + p_{\ell_1\to\ell_1+1}\cdot\mathrm{E}\left[T^{\mathrm{GR}}_{\ell_1+1,\ell_1+\ell_2}\right] + p_{\ell_1\to\ell_1-1}\cdot\mathrm{E}[T_{0\to\ell_1+\ell_2}]\right)\\ &\leq c_1c_2\cdot\left(1 + \mathrm{E}\left[T^{\mathrm{GR}}_{\ell_1+1,\ell_1+\ell_2}\right] + p_{\ell_1\to\ell_1-1}\cdot\mathrm{E}[T_{0\to\ell_1+\ell_2}]\right).\end{aligned}$$

We introduce this into (6.2), leading to

$$\begin{aligned}\mathrm{E}\left[T_{0\to\ell_1+\ell_2}\right] \leq\ & \mathrm{E}\left[T_{0\to 1}\right]+\mathrm{E}\left[T_{1\to\ell_1}\right]+\\ & c_1c_2\left(1+\mathrm{E}\left[T^{\mathrm{GR}}_{\ell_1+1,\ell_1+\ell_2}\right]+p_{\ell_1\to\ell_1-1}\cdot\mathrm{E}\left[T_{0\to\ell_1+\ell_2}\right]\right).\end{aligned}$$

Solving for $\mathrm{E}\left[T_{0\to\ell_1+\ell_2}\right]$ leads to

$$\mathrm{E}\left[T_{0\to\ell_1+\ell_2}\right] \leq \frac{\mathrm{E}\left[T_{0\to 1}\right]+\mathrm{E}\left[T_{1\to\ell_1}\right]+c_1c_2+c_1c_2\cdot\mathrm{E}\left[T^{\mathrm{GR}}_{\ell_1+1,\ell_1+\ell_2}\right]}{1-c_1c_2\cdot p_{\ell_1\to\ell_1-1}}.$$

Again by Lemma 6.4, properties (i) and (ii) of Lemma 6.3 imply that the denominator is a constant $1/c_3$. Hence,

$$\mathrm{E}\left[T_{0\to\ell_1+\ell_2}\right] \leq c_3\left(\mathrm{E}\left[T_{0\to 1}\right]+\mathrm{E}\left[T_{1\to\ell_1}\right]+c_1c_2+c_1c_2\cdot\mathrm{E}\left[T^{\mathrm{GR}}_{\ell_1+1,\ell_1+\ell_2}\right]\right). \qquad (6.5)$$

Since $\mathrm{E}\left[T_{0\to 1}\right] \leq \mathrm{E}\left[T_{1\to\ell_1}\right]$ we have that $\mathrm{E}\left[T_{0\to 1}\right]+\mathrm{E}\left[T_{1\to\ell_1}\right]=\Theta\left(\mathrm{E}\left[T_{1\to\ell_1}\right]\right)$. Now we consider the last term. Due to property (ii) of Lemma 6.3, once in ℓ_1+1 there is a constant probability of moving towards the optimum. Since the algorithm has to cover a distance of $\ell_2+\ell_1-(\ell_1+1)=\ell_2-1$, then $\mathrm{E}\left[T^{\mathrm{GR}}_{\ell_1+1,\ell_1+\ell_2}\right]=\Theta(\ell_2)$. Plugging this into (6.5) proves the claimed upper bound. □

Now we estimate the time to move from P_0 to P_{ℓ_1}. As in the previous proof, the main arguments are a recurrence relation and a Gambler's Ruin game.

Lemma 6.5. *Consider any algorithm described by Algorithm 2.3 with local mutations on* VALLEY *with* $\ell_1,\ell_2 \in \mathbb{N}\setminus\{1\}$ *and* $d_1,d_2 \in R^+$. *Then the number of relevant steps to go from the state* P_1 *to* P_{ℓ_1} *is*

$$E\left[T_{1\to\ell_1}\right]=\frac{1}{p^{\mathrm{GR}}_{1\to\ell_1}}\cdot\left(E\left[T^{\mathrm{GR}}_{1,\ell_1}\right]+\frac{p^{\mathrm{GR}}_{1\to 0}}{p_{0\to 1}}\right).$$

Proof. At the state P_1 a Gambler's Ruin game (with self-loops) occurs. The two possible outcomes are: (1) we have reached the valley P_{ℓ_1} or (2) we are back to P_0. Hence,

$$\begin{aligned}\mathrm{E}\left[T_{1\to\ell_1}\right]&=\mathrm{E}\left[T^{\mathrm{GR}}_{1,\ell_1}\right]+p^{\mathrm{GR}}_{1\to 0}\cdot\mathrm{E}\left[T_{0\to\ell_1}\right]\\ &=\mathrm{E}\left[T^{\mathrm{GR}}_{1,\ell_1}\right]+p^{\mathrm{GR}}_{1\to 0}\cdot\left(\mathrm{E}\left[T_{0\to 1}\right]+\mathrm{E}\left[T_{1\to\ell_1}\right]\right).\end{aligned}$$

Solving for $\mathrm{E}\left[T_{1\to\ell_1}\right]$ leads to

$$\mathrm{E}\left[T_{1\to\ell_1}\right] = \frac{\mathrm{E}\left[T_{1,\ell_1}^{\mathrm{GR}}\right] + p_{1\to 0}^{\mathrm{GR}} \cdot \mathrm{E}\left[T_{0\to 1}\right]}{1 - p_{1\to 0}^{\mathrm{GR}}},$$

which, by using $1 - p_{1\to 0}^{\mathrm{GR}} = p_{1\to\ell_1}^{\mathrm{GR}}$, simplifies to

$$\mathrm{E}\left[T_{1\to\ell_1}\right] = \frac{1}{p_{1\to\ell_1}^{\mathrm{GR}}} \cdot \left(\mathrm{E}\left[T_{1,\ell_1}^{\mathrm{GR}}\right] + \frac{p_{1\to 0}^{\mathrm{GR}}}{p_{0\to 1}}\right). \tag{6.6}$$

□

6.2.3 Application to SSWM

In this subsection we make use of the previous framework to analyse the SSWM for the VALLEY problem. To apply this framework we need to know how a Gambler's Ruin with the acceptance probabilities of the SSWM behaves. The following lemma contains bounds on the expected duration of the game and winning probabilities for SSWM. Although VALLEY has slopes of d_1/ℓ_1 and d_2/ℓ_2, SSWM through the action of the parameter β sees an effective gradient of $\beta \cdot d_1/\ell_1$ and $\beta \cdot d_2/\ell_2$. Varying this parameter allows the algorithm to accommodate the slope to a comfortable value. We have set this effective gradient to $\beta|\Delta f| = \Omega(1)$ so that the probability of accepting an improvement is at least a constant.

Lemma 6.6 (SSWM Gambler's Ruin)**.** *Consider a Gambler's Ruin problem as described in Theorem 6.2 with starting dollars $n_1 = 1$ and $n_2 = \ell - 1$, and probabilities p_1 and p_2 dependent on SSWM's acceptance function as follows*

$$p_1 = \frac{1}{2} \cdot p_{\mathrm{fix}}(\Delta f) \qquad p_2 = \frac{1}{2} \cdot p_{\mathrm{fix}}(-\Delta f)$$

where $\Delta f < 0$ and $(N-1)\beta|\Delta f| = \Omega(1)$. Then the winning probability of player one $P_{1\to\ell_1}^{\mathrm{GR}}$ can be bounded as follows

$$\frac{-2(N-1)\beta\Delta f}{e^{-2(N-1)\beta(n_1+n_2)\Delta f}} \leq P_{1\to\ell_1}^{\mathrm{GR}} \leq \frac{e^{-2(N-1)\beta\Delta f}}{e^{-2(N-1)\beta(n_1+n_2)\Delta f} - 1}$$

and the expected duration of the game will be $E\left[T_{1,\ell}^{\mathrm{GR}}\right] = O(1)$.

Proof. We start with the winning probability. Invoking Theorem 6.2 and simplifying the ratio of p_{fix} of symmetric fitness variations with Lemma 4.3 we obtain

$$P^{\text{GR}}_{1\to\ell_1} = \frac{1-\left(\frac{p_2}{p_1}\right)^{n_1}}{1-\left(\frac{p_2}{p_1}\right)^{n_1+n_2}} = \frac{1-\left(\frac{p_{\text{fix}}(-\Delta f)}{p_{\text{fix}}(\Delta f)}\right)^{n_1}}{1-\left(\frac{p_{\text{fix}}(-\Delta f)}{p_{\text{fix}}(\Delta f)}\right)^{n_1+n_2}} = \frac{1-e^{-2(N-1)\beta n_1\Delta f}}{1-e^{-2(N-1)\beta(n_1+n_2)\Delta f}}.$$

Notice that this is the same expression as the acceptance probability if we change β for $(N-1)\beta$ and N for ℓ. Then we can apply the bounds for the original acceptance probabilities from Lemma 4.2 to obtain the inequalities of the theorem's statement.

Finally, for the expected duration of the game we call again Theorem 6.2

$$\begin{aligned}
\mathrm{E}\left[T^{\text{GR}}_{1,\ell}\right] &= \frac{1}{p_2+p_1}\cdot\frac{n_1-(n_1+n_2)\cdot P^{\text{GR}}_{1\to\ell_1}}{p_2-p_1} \\
&\le \frac{1-\ell\cdot P^{\text{GR}}_{1\to\ell_1}}{p_2^2-p_1^2} \le \frac{1}{p_2^2-p_1^2} = \frac{1}{p_2^2\left(1-\frac{p_1^2}{p_2^2}\right)} \\
&= \frac{1}{p_2^2\left(1-e^{-4\beta(N-1)\Delta f}\right)}.
\end{aligned}$$

Note that in the last step we have used Lemma 4.3, and that since $N \ge 2$ the condition $(N-1)\beta|\Delta f| = \Omega(1)$ implies that $\beta|\Delta f| = \Omega(1)$. Hence all the parameters of SSWM's acceptance probability (Equation (4.2)) are $\Omega(1)$ and so is p_2. For the same reason the factor $1-e^{-4\beta(N-1)\Delta f}$ is constant yielding $\mathrm{E}\left[T^{\text{GR}}_{1,\ell}\right] = O(1)$. □

While the optimisation time of the (1+1) EA grows exponentially with the length of the valley, the following theorem shows that for the SSWM the growth is exponential in the depth of the valley.

Theorem 6.3. *The expected number of function evaluations $E\left[T_f\right]$ for SSWM with local mutations to reach $\mathrm{P}_{\ell_1+\ell_2}$ from P_0 on* Valley *with $\ell_1,\ell_2 \in \{2,3,\dots\}$ and $d_1,d_2 \in R^+$ is*

$$\begin{aligned}
E\left[T_f\right] &= O\left(n\cdot e^{2N\beta d_1(\ell_1+1)/\ell_1}\right) + \Theta(n\cdot\ell_2) \text{ and} \\
E\left[T_f\right] &= \Omega\left(n\cdot e^{2(N-1)\beta d_1(\ell_1-1)/\ell_1}\right) + \Theta(n\cdot\ell_2)
\end{aligned}$$

provided $\beta d_1/\ell_1$, $\beta d_2/\ell_2 = \Omega(1)$ and N being a large enough constant.

The conditions $\beta d_1/\ell_1$, $\beta d_2/\ell_2 = \Omega(1)$ are identical to those in Lemma 6.6: SSWM must have a selection strength β strong enough such that the probability of accepting a move uphill (fitness difference of d_1/ℓ_1 or d_2/ℓ_2) is $\Omega(1)$. This is a necessary and sensible condition as otherwise SSWM struggles to climb uphill (recall Theorem 5.2).

The upper and lower bounds in Theorem 6.3 are not tight because of the terms $(\ell_1+1)/\ell_1$ and $(\ell_1-1)/\ell_1$ in the exponents, respectively. However, both these terms converge to 1 as ℓ_1 grows. The running time, particularly the term $e^{2N\beta d_1(\ell_1+1)/\ell_1}$, crucially depends on βd_1, the depth of the valley after scaling. Note that the condition $\beta d_1/\ell_1 = \Omega(1)$ is equivalent to $\beta d_1 = \Omega(\ell_1)$, hence Theorem 6.3 applies if the depth after scaling is at least of the same order of growth as the length (recall that d_1 and ℓ_1 may grow with n).

Theorem 6.3 also indicates how to choose β according to the valley function in hand, in order to meet the theorem's condition and to minimise the (upper bounds on the) running time. One can always choose $\beta = \varepsilon\ell_1/d_1$ for some constant $\varepsilon > 0$ and any valley structure (even when $\ell_1 = \omega(d_1)$). This way the theorem's condition becomes $\beta d_1/\ell_1 = \varepsilon$ and the running time simplifies to $O\left(n \cdot e^{2N\beta\varepsilon(\ell_1+1)}\right) + \Theta(n \cdot \ell_2)$, where we can choose the constant $\varepsilon > 0$ as small as we like. For $N = O(1)$ we can further simplify the runtime to $O\left(n \cdot e^{O(\ell_1)}\right) + \Theta(n \cdot \ell_2)$. For all $\ell_1 \geq 2$ (and reasonable ℓ_2) this is asymptotically smaller that the expected optimisation time of the (1+1) EA, which is at least $\Omega(n^{\ell_1}) = \Omega(e^{\ell_1 \ln n})$ (see Theorem 6.1).

Proof of Theorem 6.3. The first part of the proof consists of estimating $\mathrm{E}\left[T_{1\to\ell_1}\right]$ by using the statement of Lemma 6.5. Then we will check that the conditions from Lemma 6.3 are met and we will add the $\Theta(\ell_2)$ term. Finally, we will take into account the time needed for a relevant step in the long path to obtain the n factor in the bounds (see Lemma 6.2).

As just described above we start considering $\mathrm{E}\left[T_{1\to\ell_1}\right]$ by using Lemma 6.5. Let us start with the upper bound.

$$\mathrm{E}\left[T_{1\to\ell_1}\right] = O\left(\frac{1}{p^{\mathrm{GR}}_{1\to\ell_1}} \cdot \left(\mathrm{E}\left[T^{\mathrm{GR}}_{1,\ell_1}\right] + \frac{1}{p_{0\to 1}}\right)\right).$$

Using Lemma 6.6 we bound $p^{\mathrm{GR}}_{1\to\ell_1}$ yielding

$$\mathrm{E}\left[T_{1\to\ell_1}\right] = O\left(\frac{e^{2(N-1)\beta d_1}}{2(N-1)\beta d_1/\ell_1} \cdot \left(O(1) + \frac{1}{p_{0\to 1}}\right)\right).$$

Since p_{fix} for $\Delta f < 0$ decreases when the parameters N, β and $|\Delta f|$ increase and $N\beta d_1/\ell_1 = \Omega(1)$, we get $p_{0\to 1}^{-1} = \Omega(1)$ and $O(1) + \frac{1}{p_{0\to 1}} = O\left(\frac{1}{p_{0\to 1}}\right)$. Hence,

$$\mathrm{E}\left[T_{1\to \ell_1}\right] = O\left(\frac{e^{2(N-1)\beta d_1}}{2(N-1)\beta d_1/\ell_1} \cdot \frac{1}{p_{0\to 1}}\right).$$

Using Lemma 4.2 to lower bound $p_{0\to 1}$ we get

$$\mathrm{E}\left[T_{1\to \ell_1}\right] = O\left(\frac{e^{2(N-1)\beta d_1}}{2(N-1)\beta d_1/\ell_1} \cdot \frac{e^{2N\beta d_1/\ell_1}}{2\beta \frac{d_1}{\ell_1}}\right).$$

Using $(N-1)\beta d_1/\ell_1 = \Omega(1)$ and $\beta d_1/\ell_1 = \Omega(1)$ both terms of the denominator are $\Omega(1)$ leading to

$$\mathrm{E}\left[T_{1\to \ell_1}\right] = O\left(e^{2N\beta d_1(\ell_1+1)/\ell_1}\right).$$

We now consider the lower bound. Starting again from Lemmas 6.3 and 6.5 and bounding $p_{1\to \ell_1}^{\text{GR}}$ with Lemma 6.6

$$\begin{aligned}
\mathrm{E}\left[T_{1\to \ell_1}\right] &= \Omega\left(\frac{1}{p_{1\to \ell_1}^{\text{GR}}}\right) = \Omega\left(\frac{e^{2(N-1)\beta d_1} - 1}{e^{2(N-1)\beta d_1/\ell_1}}\right) \\
&= \Omega\left(e^{2(N-1)\beta d_1 \frac{\ell_1 - 1}{\ell_1}} - \frac{1}{e^{2(N-1)\beta d_1/\ell_1}}\right) \\
&= \Omega\left(e^{2(N-1)\beta d_1 \frac{\ell_1 - 1}{\ell_1}}\right).
\end{aligned}$$

Now we need to apply Lemma 6.3 to add the $\Theta(\ell_2)$ term in both bounds. We start checking that all the conditions are satisfied. Firstly, since p_{fix} for $\Delta f > 0$ increases when the parameters (N, β and Δf) increase, then $N\beta d_2/\ell_2 = \Omega(1)$ implies $p_{\ell_1\to \ell_1+1} = \Omega(1)$. Analogously for $p_{\ell_1\to \ell_1-1}$ with $N\beta d_1/\ell_1 = \Omega(1)$ satisfying the first property. Secondly, property (ii) follows directly from Lemma 4.3 and the condition $N\beta d_2/\ell_2 = \Omega(1)$. The third property is satisfied since for $N > 1$ the acceptance probability is strictly increasing with Δf. Considering the time for a relevant step from Lemma 6.2 completes the proof. □

Finally, we graphically show the exponential scaling of SSWM's runtime on VALLEY. We fixed to constant values all the parameters but ℓ_1 which increases with the problem size with $\ell_1 = \lfloor n/10 \rfloor$. This way the term $e^{2N\beta d_1(\ell_1+1)/\ell_1}$ from Theorem 6.3 grows exponentially with n, or linearly in a logarithmic scale as in Figure 6.2.

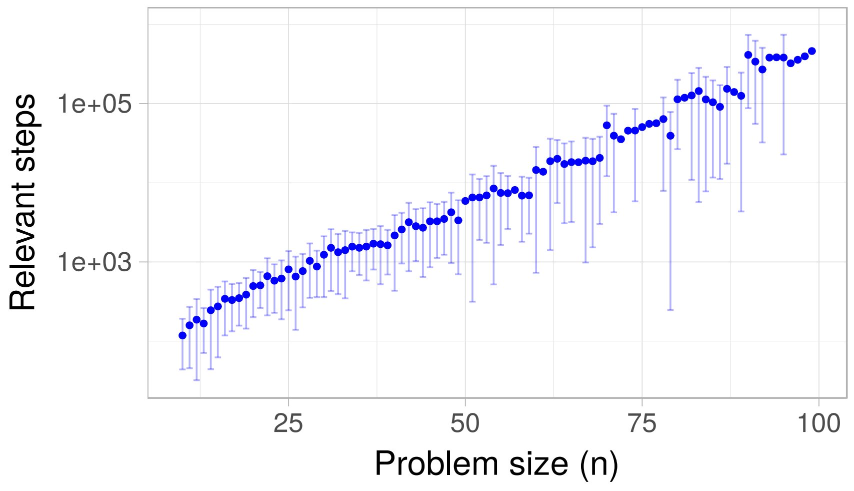


Fig. 6.2 Expected runtime of SSWM on VALLEY with $\ell_1 = \lfloor n/10 \rfloor$, $d_1 = \ell_1$, $\ell_2 = n - \ell_1$ and $d_2 = d_1 + 1$. The algorithm parameters ($N = 2$ and $\beta = 1/2$) meet the conditions required by Theorem 6.3. A logarithmic scale with base 10 is used for the y-axis. Results were averaged over 50 independent runs and the error bars contain $\pm$ one standard deviation.

6.2.4 Application to the Metropolis algorithm

We now apply the framework from Section 6.2.2 to the Metropolis algorithm. Since the analysis follows very closely the one of SSWM the proofs for this subsection are provided in Appendix 6.B. We first cast MA on VALLEY as a Gambler's Ruin problem. Like SSWM, MA can make use of its parameter α to accommodate the gradient of VALLEY.

Lemma 6.7 (Metropolis Gambler's Ruin downhill)**.** *Consider a Gambler's Ruin problem as described in Theorem 6.2 with starting dollars $n_1 = 1$ and $n_2 = \ell - 1$, and probabilities p_1 and p_2 dependent on MA's acceptance function as follows*

$$p_1 = \frac{1}{2} \cdot e^{-\alpha \Delta f} \qquad p_2 = \frac{1}{2}$$

where $\Delta f < 0$ and $\alpha|\Delta f| = \Omega(1)$. Then the winning probability of player one P_1 can be bounded as follows

$$\frac{-\alpha \Delta f}{e^{-\alpha \ell \Delta f}} < P_1^{\mathrm{GR-Met}} < \frac{e^{-\alpha \Delta f}}{e^{-\alpha \ell \Delta f} - 1}$$

and the expected duration of the game will be $E\left[T_{1,\ell}^{\mathrm{GR}}\right] = O(1)$.

Lastly, we make use of the previous lemma and the framework presented in Section 6.2.2 to determine bounds on the runtime of the Metropolis algorithm for VALLEY. Note that the required conditions are similar to those from Theorem 6.3 for the SSWM algorithm, with only difference being that the parameter α substitutes the selection strength β. Hence the previous considerations for SSWM translate to MA on VALLEY by simply applying $\beta \leftarrow \alpha$.

Theorem 6.4. *The expected number of function evaluations $E\left[T_f\right]$ for the Metropolis algorithm to reach $P_{\ell_1+\ell_2}$ from P_0 on* VALLEY *with $\ell_1, \ell_2 \in \mathbb{N} \setminus \{1\}$ and $d_1, d_2 \in R^+$ is*

$$E(T_f) = O\left(n \cdot e^{\alpha d_1 (1+1/\ell_1)}\right) + \Theta(n \cdot \ell_2) \text{ and}$$
$$E(T_f) = \Omega\left(n \cdot e^{\alpha d_1 (1-1/\ell_1)}\right) + \Theta(n \cdot \ell_2)$$

provided $\alpha d_1/\ell_1$, $\alpha d_2/\ell_2 = \Omega(1)$.

Finally, we recover the experimental setting from Figure 6.2 to graphically show the exponential scaling of MA's runtime on VALLEY.

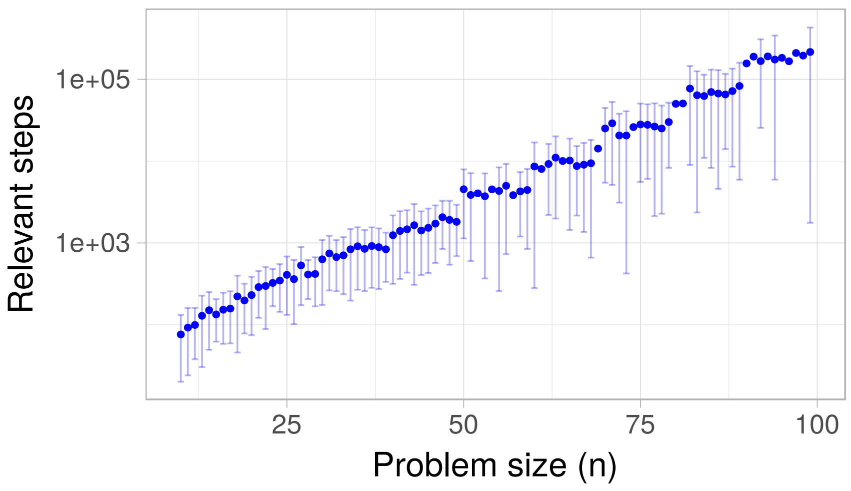


Fig. 6.3 Expected runtime of MA on VALLEY with $\ell_1 = \lfloor n/10 \rfloor$, $d_1 = \ell_1$, $\ell_2 = n - \ell_1$ and $d_2 = d_1 + 1$. The algorithm parameters ($\alpha = 1$) meet the conditions required by Theorem 6.4. A logarithmic scale with base 10 is used for the y-axis. Results were averaged over 50 independent runs and the error bars contain $\pm$ one standard deviation.

6.3 Crossing Concatenated Valleys

We define a class of functions called VALLEYPATH consisting of m consecutive valleys of the same size. Each of the consecutive valleys is shifted such that the fitness at the beginning of each valley is the same as that at the end of the previous valley (see Figure 6.4). Fitness values from one valley to the next valley increase by an amount of $d_2 - d_1 > 0$. Formally:

Definition 6.4 (VALLEYPATH).

$$h(i,j)_{\text{VALLEYPATH}} := \begin{cases} j\cdot(d_2-d_1)+d_1-i\cdot\frac{d_1}{\ell_1} & \text{if } i\le\ell_1 \\ j\cdot(d_2-d_1)+(i-\ell_1)\cdot\frac{d_2}{\ell_2} & \text{if } \ell_1<i\le\ell. \end{cases}$$

Here $0<j\le m$ indicates a valley while $0\le i\le\ell_1+\ell_2=\ell$ indicates the position in the given valley. Hence, the global optimum is the path point $P_{m\cdot\ell}$.

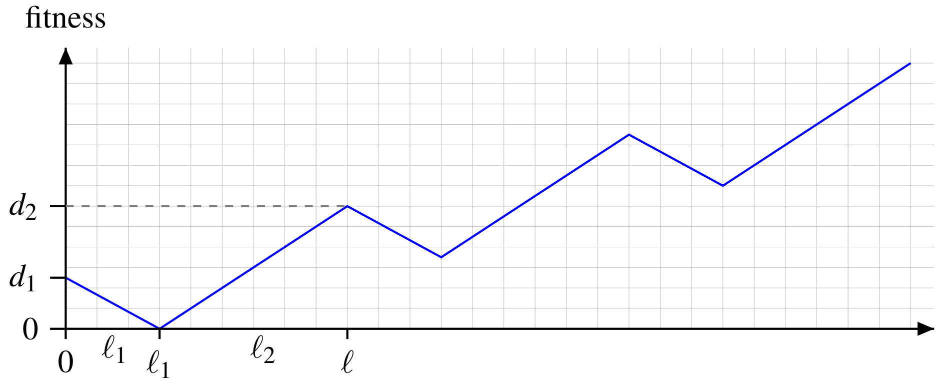


Fig. 6.4 Sketch of the function VALLEYPATH.

VALLEYPATH represents a rugged fitness landscape with many valleys and many local optima (peaks). It loosely resembles a “big valley” structure found in many real-world problems (Boese et al., 1994; Merz and Freisleben, 1998; Ochoa and Veerapen, 2016; Reeves, 1999): from a high-level view the concatenation of valleys indicates a “global” gradient, i. e. the direction towards valleys at higher indices. The difficulty for optimisation algorithms is to overcome these many local optima and to still be able to identify the underlying gradient. We show here that both SSWM and MA are able to exploit this global gradient and find the global optimum efficiently. Note that VALLEYPATH is a very broad function class in that it allows for many shapes to emerge, from few deep valleys to many shallow ones. Our results hold for all valley paths with $d_1<d_2$.

As in the analysis for VALLEY, instead of considering the whole Markov chain underlying VALLEYPATH we take a high-level view and consider the chain that describes transitions between neighbouring peaks. Since the peaks have increasing fitness, this chain is quite simple and allows for an easy application of drift arguments. By choosing the number of peaks to the right of the current peak as distance function, the next theorem shows that, if we can find constant bounds for the drift, we will only need to repeat the VALLEY experiment for as many peaks as there are in VALLEYPATH.

Theorem 6.5. *Consider any algorithm described by Algorithm 2.3 with local mutations on* VALLEYPATH. *Consider the points in time where the algorithm is on a peak and focus on* *transitions between different peaks. Let X_t be a random variable describing the number of peaks to the right of the current valley at the t-th time a different peak is reached. If the drift over peaks Δ can be lower bounded by some positive constant*

$$\Delta := E[X_t - X_{t+1} \mid X_t > 0] \geq c > 0 \tag{6.7}$$

then the expected number of function evaluations $E\left[T_f\right]$ to reach the optimum starting from any peak is

$$E\left[T_f\right] = O\left(m \cdot E\left[T^{O}_{\text{VALLEY}}\right]\right) \text{ and } \Omega\left(m \cdot E\left[T^{\Omega}_{\text{VALLEY}}\right]\right)$$

where m is the number of valleys that compose VALLEYPATH, *and $E\left[T^{O}_{\text{VALLEY}}\right]$ and $E\left[T^{\Omega}_{\text{VALLEY}}\right]$ are the upper and lower bounds for* VALLEY *respectively.*

Proof. The lower bound is trivial since the algorithm can only move to a neighbour peak and has to visit m peaks. The upper bound follows from the standard additive drift theorem (Theorem 2.7). □

To compute the drift over the peaks Δ from Equation (6.7) we perform a slightly different abstraction over the VALLEYPATH problem. We will also consider, apart from the peaks (local maxima), the points of minimal fitness between them (valleys). For simplicity we will use the following notation.

Definition 6.5 (VALLEYPATH Notation)**.** *Consider any algorithm described by Algorithm 2.3 with local mutations where the current search point is any extreme point (a maximum or minimum) of* VALLEYPATH. *If the algorithm is on a valley (minimum) we will denote by:*

(1) T_{peaks} the first hitting time of either of the neighbouring peaks,

(2) $p_r^{\uparrow}$ the probability of the algorithm being in the right-hand peak at T_{peaks},

(3) $p_l^{\uparrow} = 1 - p_r^{\uparrow}$ the probability of the algorithm being in the left-hand peak at T_{peaks}.

If the algorithm is on a peak (a maximum) we will denote by:

(4) $T_{\min}$ the first hitting time of either of the neighbouring minima,

(5) $p_r^{\downarrow}$ the probability of the algorithm being in the right-hand minimum at $T_{\min}$,

(6) $p_l^{\downarrow} = 1 - p_r^{\downarrow}$ the probability of the algorithm being in the left-hand minimum at $T_{\min}$.

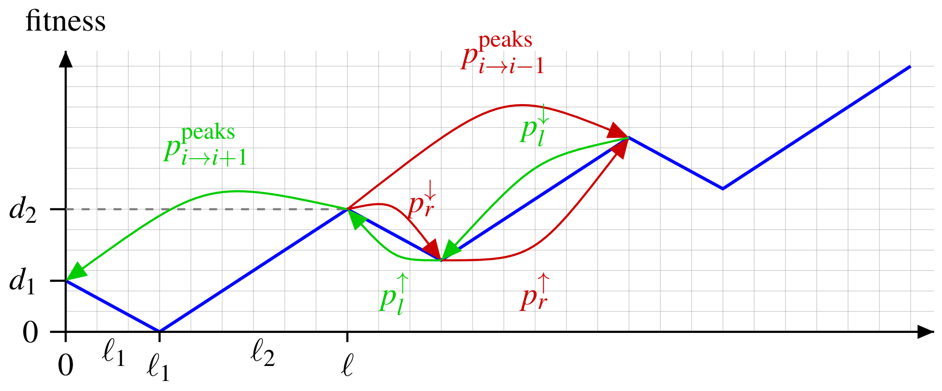


Fig. 6.5 Sketch of the function ValleyPath with the probabilities used for its runtime analysis.

The following lemma computes the drift between peaks Δ from Equation (6.7) by introducing transition probabilities between neighbouring peaks $p^{\text{peaks}}_{i\to i-1}$ and $p^{\text{peaks}}_{i\to i+1}$ (see Figure 6.5). These two new probabilities can be expressed in terms of the transition probabilities between consecutive peaks and minima from Definition 6.5, yielding a neat expression for the drift.

Lemma 6.8. *In the context of Theorem 6.5 and Definition 6.5, if $p_r^{\downarrow}p_r^{\uparrow} \geq \gamma p_l^{\downarrow}p_l^{\uparrow}$ for some constant $\gamma > 1$ then the drift between peaks will be $\Omega(1)$.*

Proof. Let us start from the definition of the drift between peaks from Theorem 6.5, we expand the drift in terms of two probabilities: $p^{\text{peaks}}_{i\to i-1}$ which reads as the probability of decreasing the number of peaks on the right by 1 and for the opposite event $p^{\text{peaks}}_{i\to i+1} = 1 - p^{\text{peaks}}_{i\to i-1}$. Then,

$$\begin{aligned}\mathrm{E}\left[X_t - X_{t+1} \mid X_t = i > 0\right] &= i - p^{\text{peaks}}_{i\to i-1}(i-1) - p^{\text{peaks}}_{i\to i+1}(i+1)\\ &= i\cdot\left(1 - p^{\text{peaks}}_{i\to i-1} - p^{\text{peaks}}_{i\to i+1}\right) + p^{\text{peaks}}_{i\to i-1} - p^{\text{peaks}}_{i\to i+1}\\ &= 2p^{\text{peaks}}_{i\to i-1} - 1\end{aligned}$$

where in the last step we have used $p^{\text{peaks}}_{i\to i-1} + p^{\text{peaks}}_{i\to i+1} = 1$. Therefore a sufficient condition for the drift to be constant is $p^{\text{peaks}}_{i\to i-1} \geq \frac{1}{2} + \frac{1}{c}$ for some constant $c \geq 0$.

We can break down this term using the four probabilities from Definition 6.5. We consider the Markov Chain composed of the extreme points (maxima and minima) of ValleyPath and the algorithm on a peak. After two steps the system can be only in one of three points: with probability $p_l^{\downarrow}p_l^{\uparrow}$ the algorithm will reach the peak on its left, analogously it will reach the right peak with probability $p_r^{\downarrow}p_r^{\uparrow}$ and with the remaining probability $p_r^{\downarrow}p_l^{\uparrow} + p_l^{\downarrow}p_r^{\uparrow}$ the algorithm will leave and return to the starting peak before reaching any other peak. We can now express the probability of moving to a specific peak given that we have moved to a peak

$p_{i\to i-1}^{\text{peaks}}$ as

$$p_{i\to i-1}^{\text{peaks}} = \frac{p_r^{\downarrow} p_r^{\uparrow}}{p_r^{\downarrow} p_r^{\uparrow} + p_l^{\downarrow} p_l^{\uparrow}} = \frac{1}{1 + \frac{p_l^{\downarrow} p_l^{\uparrow}}{p_r^{\downarrow} p_r^{\uparrow}}}.$$

The previous condition $p_{i\to i-1}^{\text{peaks}} \geq \frac{1}{2} + \frac{1}{c}$ can now be translated to $p_r^{\downarrow} p_r^{\uparrow} \geq \gamma p_l^{\downarrow} p_l^{\uparrow}$ for some constant $\gamma > 1$. □

The previous lemma gives us a simple equation that determines the drift. Losing rigour for a moment we neglect the factor γ and identify some regimes where the overall drift is positive: (1) $p_r^{\downarrow} > p_l^{\downarrow}$ and $p_r^{\uparrow} > p_l^{\uparrow}$, (2) $p_r^{\downarrow} \gg p_l^{\downarrow}$ and $p_r^{\uparrow} < p_l^{\uparrow}$ or (3) $p_r^{\downarrow} < p_l^{\downarrow}$ and $p_r^{\uparrow} \gg p_l^{\uparrow}$. In the next lemma we recover the original Markov Chain and express these four probabilities in terms of the real transition probabilities between the states of VALLEYPATH. We will also make an assumption on the algorithm (the probability of accepting an improvement of Δf must be exponentially bigger than the probability of accepting a worsening of the same size with Δf). As the reader might notice from the previous section both SSWM and MA meet this condition.

Finally we can simplify the condition to have a positive drift in a neat expression that only depends on the depth of the valleys (d_1 and d_2) and the parameter λ of the acceptance probability distribution.

Lemma 6.9. *In the context of Lemma 6.8, consider any algorithm described by Algorithm 2.3 with an acceptance function such that* $\frac{p_{\text{acc}}(\Delta f)}{p_{\text{acc}}(-\Delta f)} = e^{\lambda \Delta f}$ *for some* $\lambda \in \mathbb{R}^+$. *Then*

$$\frac{p_l^{\downarrow} p_l^{\uparrow}}{p_r^{\downarrow} p_r^{\uparrow}} = e^{-\lambda(d_2 - d_1)}.$$

Proof. According to our notation (Definition 6.5) $p_l^{\downarrow}$ reads as the probability of reaching the minimum on the left of a peak before reaching the minimum on its right. As in the VALLEY section we can break down the process of moving to a neighbouring minimum in two steps: (1) first moving just one point towards the left slope and (2) winning a Gambler's Ruin game starting with one dollar, using a notation in the same spirit as in the previous section (see Definition 6.3). We will respectively denote the probability of the events (1) and (2) $p_{\ell_2}^{\downarrow}$ and $p_{\ell_2}^{\text{GR}\downarrow}$, where ℓ_2 determines that the process is on the slope with length ℓ_2. Using the same rationale for the other probabilities we can rewrite the quotient from Lemma 6.8 as

$$\begin{aligned}
\frac{p_l^{\downarrow}p_l^{\uparrow}}{p_r^{\downarrow}p_r^{\uparrow}} &= \frac{\left(\frac{p_{\ell_2}^{\downarrow}p_{\ell_2}^{\mathrm{GR}\downarrow}}{p_{\ell_2}^{\downarrow}p_{\ell_2}^{\mathrm{GR}\downarrow}+p_{\ell_1}^{\downarrow}p_{\ell_1}^{\mathrm{GR}\downarrow}}\right)\cdot\left(\frac{p_{\ell_1}^{\uparrow}p_{\ell_1}^{\mathrm{GR}\uparrow}}{p_{\ell_1}^{\uparrow}p_{\ell_1}^{\mathrm{GR}\uparrow}+p_{\ell_2}^{\uparrow}p_{\ell_2}^{\mathrm{GR}\uparrow}}\right)}{\left(\frac{p_{\ell_1}^{\downarrow}p_{\ell_1}^{\mathrm{GR}\downarrow}}{p_{\ell_1}^{\downarrow}p_{\ell_1}^{\mathrm{GR}\downarrow}+p_{\ell_2}^{\downarrow}p_{\ell_2}^{\mathrm{GR}\downarrow}}\right)\cdot\left(\frac{p_{\ell_2}^{\uparrow}p_{\ell_2}^{\mathrm{GR}\uparrow}}{p_{\ell_1}^{\uparrow}p_{\ell_1}^{\mathrm{GR}\uparrow}+p_{\ell_2}^{\uparrow}p_{\ell_2}^{\mathrm{GR}\uparrow}}\right)} \\
&= \frac{p_{\ell_2}^{\downarrow}p_{\ell_2}^{\mathrm{GR}\downarrow}\cdot p_{\ell_1}^{\uparrow}p_{\ell_1}^{\mathrm{GR}\uparrow}}{p_{\ell_1}^{\downarrow}p_{\ell_1}^{\mathrm{GR}\downarrow}\cdot p_{\ell_2}^{\uparrow}p_{\ell_2}^{\mathrm{GR}\uparrow}} \\
&= \frac{p_{\ell_2}^{\downarrow}}{p_{\ell_2}^{\uparrow}}\cdot\frac{p_{\ell_1}^{\uparrow}}{p_{\ell_1}^{\downarrow}}\cdot\frac{p_{\ell_2}^{\mathrm{GR}\downarrow}}{p_{\ell_2}^{\mathrm{GR}\uparrow}}\cdot\frac{p_{\ell_1}^{\mathrm{GR}\uparrow}}{p_{\ell_1}^{\mathrm{GR}\downarrow}}.
\end{aligned}$$

Using the Gambler's Ruin problem results (see Theorem 2.3) we can expand the previous equality into

$$\frac{p_l^{\downarrow}p_l^{\uparrow}}{p_r^{\downarrow}p_r^{\uparrow}} = \frac{p_{\ell_2}^{\downarrow}}{p_{\ell_2}^{\uparrow}}\cdot\frac{p_{\ell_1}^{\uparrow}}{p_{\ell_1}^{\downarrow}}\cdot\frac{\left(\frac{1-\frac{p_{\ell_2}^{\uparrow}}{p_{\ell_2}^{\downarrow}}}{1-\left(\frac{p_{\ell_2}^{\uparrow}}{p_{\ell_2}^{\downarrow}}\right)^{\ell_2}}\right)}{\left(\frac{1-\frac{p_{\ell_2}^{\downarrow}}{p_{\ell_2}^{\uparrow}}}{1-\left(\frac{p_{\ell_2}^{\downarrow}}{p_{\ell_2}^{\uparrow}}\right)^{\ell_2}}\right)}\cdot\frac{\left(\frac{1-\frac{p_{\ell_1}^{\downarrow}}{p_{\ell_1}^{\uparrow}}}{1-\left(\frac{p_{\ell_1}^{\downarrow}}{p_{\ell_1}^{\uparrow}}\right)^{\ell_1}}\right)}{\left(\frac{1-\frac{p_{\ell_1}^{\uparrow}}{p_{\ell_1}^{\downarrow}}}{1-\left(\frac{p_{\ell_1}^{\uparrow}}{p_{\ell_1}^{\downarrow}}\right)^{\ell_1}}\right)}.$$

Now using the acceptance function's property from the lemma statement we can simplify this expression to

$$\frac{p_l^{\downarrow}p_l^{\uparrow}}{p_r^{\downarrow}p_r^{\uparrow}} = e^{-\lambda d_2/\ell_2}\cdot e^{\lambda d_1/\ell_1}\cdot\frac{\left(\frac{1-e^{\lambda d_2/\ell_2}}{1-e^{\lambda d_2}}\right)}{\left(\frac{1-e^{-\lambda d_2/\ell_2}}{1-e^{-\lambda d_2}}\right)}\cdot\frac{\left(\frac{1-e^{-\lambda d_1/\ell_1}}{1-e^{-\lambda d_1}}\right)}{\left(\frac{1-e^{\lambda d_1/\ell_1}}{1-e^{\lambda d_1}}\right)}.$$

Notice that the last four terms have the same format as the equation for p_{fix} (with different parameters). Lets rename them as p_{fix}^{*} for simplicity

$$\frac{p_l^{\downarrow}p_l^{\uparrow}}{p_r^{\downarrow}p_r^{\uparrow}} = e^{-\lambda d_2/\ell_2}\cdot e^{\lambda d_1/\ell_1}\cdot\left(\frac{p_{\text{fix}}^{*\downarrow\ell_2}}{p_{\text{fix}}^{*\uparrow\ell_2}}\right)\cdot\left(\frac{p_{\text{fix}}^{*\uparrow\ell_1}}{p_{\text{fix}}^{*\downarrow\ell_1}}\right).$$

We find again the ratio between symmetric fitness differences, then Lemma 4.3 highly simplifies the previous expression to

$$\frac{p_l^{\downarrow} p_l^{\uparrow}}{p_r^{\downarrow} p_r^{\uparrow}} = e^{-\lambda d_2/\ell_2} \cdot e^{\lambda d_1/\ell_1} \cdot e^{-\lambda d_2(\ell_2-1)/\ell_2} \cdot e^{\lambda d_1(\ell_1-1)/\ell_1} = e^{-\lambda(d_2-d_1)}. \qquad \square$$

6.3.1 Application for SSWM and the Metropolis algorithm

In the next two theorems we apply the previous results on VALLEYPATH to the SSWM and Metropolis algorithms. The application is straightforward when making the parameter $\lambda = \Omega(1)$ and the depths of the valley (d_1 and d_2) differ in some positive constant. Notice that it could be the case that $d_2 - d_1$ is smaller than a constant but the parameters are big enough to compensate for this effect and still have a positive drift over peaks. However this increase in the parameters will affect the optimisation time between peaks (i.e. the VALLEY problem). Note that, by applying Theorem 6.1, it is easy to see that the runtime of the (1+1) EA will be exponential in the length of the individual valleys, hence the algorithm will be efficient only for valley paths consisting of valleys of moderate length.

The remaining conditions that Theorems 6.6 and 6.7 require are those already required on the analysis for VALLEY (see Theorems 6.3 and 6.4).

Theorem 6.6. *The expected number of function evaluations $E[T_f]$ for SSWM to reach the optimum starting from any peak on* VALLEYPATH *with $2\beta(N-1)\cdot(d_2-d_1) \geq c$ for some constant $c > 0$ is*

$$E[T_f] = O\left(m \cdot n \cdot \left(e^{2N\beta d_1(l_1+1)/l_1} + \Theta(l_2)\right)\right) \text{ and}$$
$$E[T_f] = \Omega\left(m \cdot n \cdot \left(e^{2(N-1)\beta d_1(l_1-1)/l_1} + \Theta(l_2)\right)\right)$$

provided $\ell_1, \ell_2 \in \mathbb{N} \setminus \{1\}$, $d_2 > d_1$, $N = \Omega(1)$ and $\beta d_1/\ell_1, \beta d_2/\ell_2 = \Omega(1)$.

Proof. Due to Lemma 4.3, SSWM meets the exponential ratio property needed by Lemma 6.9 with $\lambda = 2\beta(N-1)$. Then we can say that

$$\frac{p_l^{\downarrow} p_l^{\uparrow}}{p_r^{\downarrow} p_r^{\uparrow}} = e^{-\lambda(d_2-d_1)} = \frac{1}{e^{2\beta(N-1)(d_2-d_1)}} = \frac{1}{\gamma}.$$

Since $2\beta(N-1)(d_2-d_1) \geq c > 0$, then γ is a constant greater than 1 fulfilling the condition required by Lemma 6.8 for the drift to be constant. Finally we apply Theorem 6.5 taking into account the optimisation time for VALLEY to obtain the claimed result. $\square$

An equivalent result to that of the SSWM for VALLEYPATH is shown for MA in the following theorem.

Theorem 6.7. *The expected number of function evaluations $E\left[T_f\right]$ for the Metropolis algorithm to reach the optimum starting from any peak on* VALLEYPATH *with $\alpha(d_2 - d_1) \geq c$ for some constant $c > 0$ is*

$$E\left[T_f\right] = O\left(m \cdot n \cdot \left(e^{\alpha d_1 (l_1+1)/l_1} + \Theta(l_2)\right)\right) \text{ and}$$
$$E\left[T_f\right] = \Omega\left(m \cdot n \cdot \left(e^{\alpha d_1 (l_1-1)/l_1} + \Theta(l_2)\right)\right)$$

provided $\ell_1, \ell_2 \in \mathbb{N} \setminus \{1\}$, $d_2 > d_1$, and $\alpha d_1/\ell_1, \alpha d_2/\ell_2 = \Omega(1)$.

Proof. The proof follows exactly as the proof of Theorem 6.6 with the only difference that $\lambda = \alpha$ (see Equation (2.3)). □

Note that our approach can be extended to concatenations of valleys of different sizes, assuming $d_1 < d_2$ for each valley. In this case the expression of the runtime would be dominated by the deepest valley.

Finally, we graphically represent the runtime of SSWM and MA on VALLEYPATH. In this case the x-axis of Figure 6.6 represents the number of valleys that constitutes VALLEYPATH, with each of these valleys having constant values for its four parameters (ℓ_1, ℓ_2, d_1 and d_2). We can observe, as proven in Theorems 6.6 and 6.7, that both algorithms solve this problem in linear time.

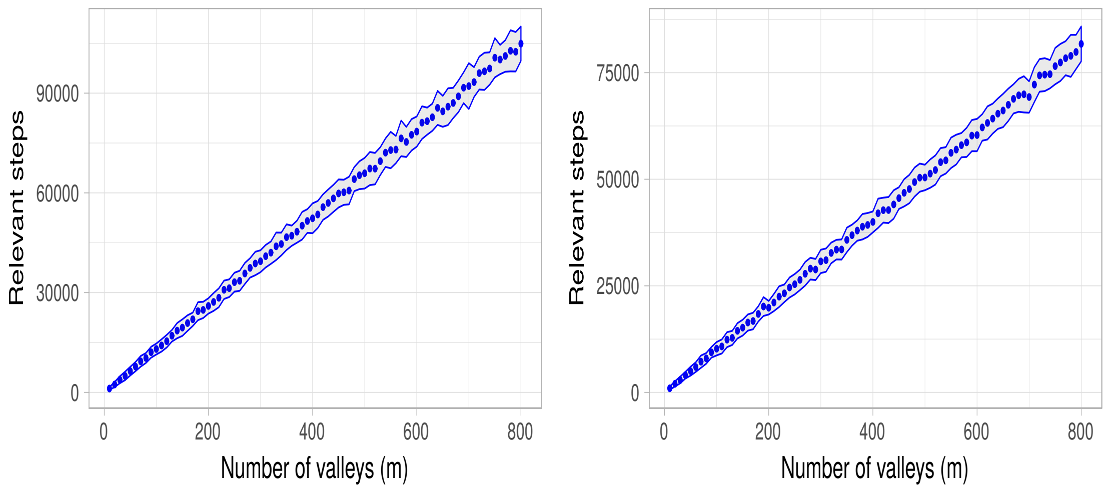


Fig. 6.6 Expected runtime of SSWM (left) and MA (right) on VALLEYPATH. Results were averaged over 50 independent runs and the shadowed zones include ± one standard deviation.

6.4 Global Mutations Speed-up Sharp Cliff Optimisation

This section is based on the following publications:

1. Paixão, T., Pérez Heredia, J., Sudholt, D., and Trubenová, B. (2015). First steps towards a runtime comparison of natural and artificial evolution. *Proceedings of the Genetic and Evolutionary Computation Conference 2015 (GECCO '15)*, pages 1455–1462. ACM.

2. Paixão, T., Pérez Heredia, J., Sudholt, D., and Trubenová, B. (2017). Towards a runtime comparison of natural and artificial evolution. *Algorithmica*, 78(2):681–713.

We conclude this chapter by using the, more common, encoding from functions of unitation (i.e., the fitness is determined by the number of ones in the bitstring). On top of this encoding, we will define a sharp VALLEY-like function resembling a deep cliff. Note that this is a really hard problem for SSWM, since the depth was the critical parameter in Theorem 6.3. However, we will allow SSWM to use global mutations (see Definition 2.6). Although this is a slightly deviation from the *weak mutation* condition from the SSWM regime (see Subsection 3.2.3), we believe that is highly interesting from an algorithmic point of view. We will show that by combining global mutations with non-elitism SSWM manages to escape a hard local optima more efficiently than elitist algorithms such as the (1+1) EA.

Algorithm 6.1: SSWM with global mutations

```
1  Initialise $x \in \{0,1\}^n$
2  Choose a population size $N \in \mathbb{N}$
3  Choose a scaling parameter $\beta \in \mathbb{R}^+$
4  repeat
5      $y \leftarrow$ flip each bit of $x$ uniform at random with probability $1/n$
6      $\Delta f \leftarrow f(y) - f(x)$
7      Choose $r \in [0,1]$ uniformly at random
8      if $r \leq p_{\text{fix}}(\Delta f)$ then
9          $x \leftarrow y$
10 until stop;
```

Jansen and Wegener (2007) compared the ability of the (1+1) EA and a Metropolis algorithm in crossing fitness valleys and found that both showed similar performance on *smooth integer* functions: functions where two Hamming neighbours have a fitness difference of at most 1 (Jansen and Wegener, 2007, Section 6).

We consider a similar function, generalising a construction by Jägersküpper and Storch (2007): the function CLIFF_d is defined such that non-elitist algorithms have a chance to jump down a "cliff" of height roughly d and to traverse a fitness valley of Hamming distance d to the optimum (see Figure 6.7). Note that d may depend on n.

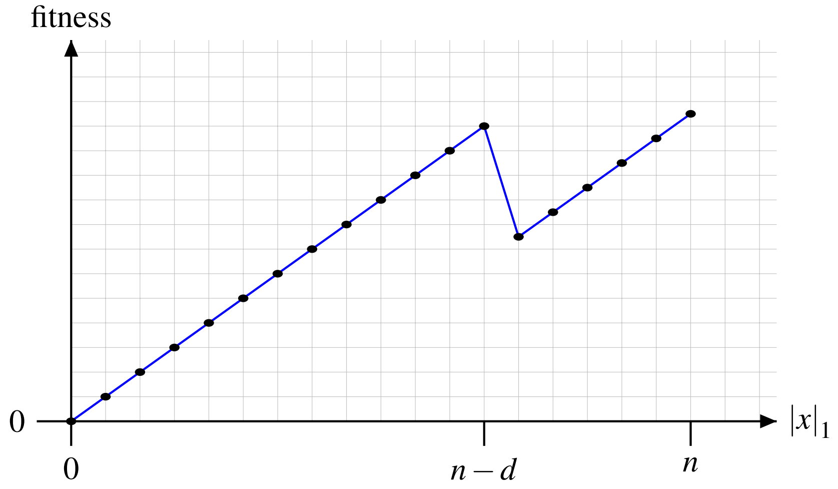


Fig. 6.7 Sketch of the function CLIFF_d.

Definition 6.6 (Cliff).

$$\text{CLIFF}_d(x) = \begin{cases} |x|_1 & \textit{if } |x|_1 \leq n-d \\ |x|_1 - d + \frac{1}{2} & \textit{otherwise} \end{cases}$$

where $|x|_1 = \sum_{i=1}^{n} x_i$ *counts the number of ones.*

The (1+1) EA typically optimises CLIFF_d through a direct jump from the top of the cliff to the optimum, which takes expected time $\Theta(n^d)$.

Theorem 6.8. *The expected optimisation time of the (1+1) EA on* CLIFF_d*, for* $2 \leq d \leq n/2$*, is* $\Theta(n^d)$.

In order to prove Theorem 6.8, the following lemma will be useful for showing that the top of the cliff is reached with good probability. More generally, it shows that the conditional probability of increasing the number of ones in a search point to j, given it is increased to some value of j or higher, is at least $1/2$.

Lemma 6.10. *For all* $0 \leq x < j \leq n$,

$$\frac{\text{mut}(x, j)}{\sum_{k=j}^{n} \text{mut}(x, k)} \geq \frac{1}{2}.$$

The proof of this lemma is presented in Appendix 6.C.

Proof of Theorem 6.8. From any search point with $x < n-d$ ones, the probability of reaching a search point with higher fitness is at least $\frac{n-x}{en}$. The expected time for accepting a search point with at least $n-d$ ones is at most $\sum_{x=0}^{n-d-1} \frac{en}{n-x} = O(n \log n)$. Note that this is $O(n^d)$ since $d \geq 2$.

We claim that with probability $\Omega(1)$, the first such search point has $n-d$ ones: with probability at least $1/2$ the initial search point will have at most $n-d$ ones. Invoking Lemma 6.10 with $j := n-d$, with probability at least $1/2$ the top of the cliff is reached before any other search point with at least $n-d$ ones.

Once on the top of the cliff the algorithm has to jump directly to the optimum to overcome it. The probability of such a jump is $\frac{1}{n^d}\left(1-\frac{1}{n}\right)^{n-d}$ and therefore the expected time to make this jump is $\Theta(n^d)$. □

SSWM with global mutations also has an opportunity to make a direct jump to the optimum. However, compared to the (1+1) EA its performance slightly improves when considering shorter jumps and accepting a search point of inferior fitness. The following theorem shows that for large enough cliffs, $d = \omega(\log n)$, the expected optimisation time is by a factor of $e^{\Omega(d)} = n^{\omega(1)}$ smaller than that of the (1+1) EA. Although both algorithms need a long time for large d, the speedup of SSWM is significant for large d.

Theorem 6.9. *The expected optimisation time of SSWM with global mutations and $\beta = 1, N = \frac{1}{2}\ln(9n)$ on* CLIFF_d *with $d = \omega(\log n)$ is at most $n^d / e^{\Omega(d)}$.*

Proof. We define R as the expected time for reaching a search point with either $n-d$ or n ones, when starting with a worst possible non-optimal search point. Let T_{cliff} be the random optimisation time when starting with any search point of $n-d$ ones, hereinafter called the *top of the cliff* or a *local peak*. Then the expected optimisation time from any initial point is at most $R + \mathrm{E}[T_{\text{cliff}}]$.

Let p_{success} be the probability that SSWM starting on top of the cliff will reach the optimum before reaching the top of the cliff again. We call the time period needed to reach the top of the cliff, or the global optimum, a *trial*. After the end of a trial, taking at most R expected generations, with probability $1 - p_{\text{success}}$ SSWM returns to the top of the cliff again, so

$$\mathrm{E}[T_{\text{cliff}}] \leq R + (1-p_{\text{success}}) \cdot \mathrm{E}[T_{\text{cliff}}] \Leftrightarrow \mathrm{E}[T_{\text{cliff}}] \leq \frac{R}{p_{\text{success}}}. \tag{6.8}$$

We first bound the worst-case time to return to the local peak or a global optimum as $R = O(n \log n)$. Let S_1 be the set of all search points with at most $n-d$ ones and $S_2 := \{0,1\}^n \setminus S_1$.

As long as the current search point remains within S_2, SSWM behaves like on ONEMAX. Here we have $2N\beta = \ln(9n)$ and by

$$2(N-1)\beta \geq \ln(cn) \Leftrightarrow 2N\beta \geq \ln\left(e^{2\beta}cn\right),$$

along with $e^{2\beta} \cdot c = e^2 \cdot c < 9$ for a suitable constant $c > 1.2$ (e. g. $c := 1.21$)[1], the condition $2(N-1)\beta \geq \ln(cn)$ of Theorem 5.1 is satisfied. Repeating arguments from the proof of Theorem 5.1, in expected time $O\left(n \log n \cdot \left(1 + \frac{1}{2\beta}\right)\right) = O(n \log n)$ (as here $\beta = 1$) SSWM either finds a global optimum or a search point in S_1. Likewise, as long as the current search point remains within S_1, SSWM essentially behaves like on ONEMAX and within expected time $O(n \log n)$ either the top of the cliff or a search point in S_2 is found.

SSWM can switch indefinitely between S_1 and S_2 within one trial, as long as no optimum nor a local peak is reached. Let X_t be the number of ones at time t, then the conditional probability of moving to a local peak—when from a search point with $x < n-d$ ones either a local peak or a non-optimal search point in S_2 is reached—is

$$\begin{aligned}
&\Pr\left(X_{t+1} = n-d \mid X_{t+1} \geq n-d, X_t = x < n-d\right) \\
&= \frac{\Pr\left(X_{t+1} = n-d \mid X_t = x < n-d\right)}{\Pr\left(X_{t+1} \geq n-d \mid X_t = x < n-d\right)} \\
&= \frac{\mathrm{mut}(x, n-d) \cdot p_{\mathrm{fix}}(n-d-x)}{\mathrm{mut}(x, n-d) \cdot p_{\mathrm{fix}}(n-d-x) + \sum_{k=n-d+1}^{n-1} \mathrm{mut}(x,k) \cdot p_{\mathrm{fix}}(k-x-d+1/2)} \\
&= \frac{\mathrm{mut}(x, n-d)}{\mathrm{mut}(x, n-d) + \sum_{k=n-d+1}^{n-1} \mathrm{mut}(x,k) \cdot p_{\mathrm{fix}}(k-x-d+1/2)/p_{\mathrm{fix}}(n-d-x)} \\
&\geq \frac{\mathrm{mut}(x, n-d)}{\sum_{k=n-d}^{n} \mathrm{mut}(x,k)}
\end{aligned}$$

as $p_{\mathrm{fix}}(n-d-x) \geq p_{\mathrm{fix}}(k-x-d+1/2)$ for all $n-d < k < n$. By Lemma 6.10, the above fraction is at least $1/2$. Hence every time SSWM increases the number of ones to at least $n-d$, with probability at least $1/2$ a local peak is found. This means that in expectation at most two transitions from S_1 to S_2 will occur before a local peak is reached, and the overall expected time spent in S_1 and S_2 is at most $R = O(1) \cdot O(n \log n)$.

The remainder of the proof now shows a lower bound on p_{success}, the probability of a trial being successful. A sufficient condition for a successful trial is that the following events occur: the next mutation creates a search point with $n-d+k$ ones, for some integer

[1]The reason for $c > 1.2$ and not 1 as in Theorem 5.1 is that here global mutations are used (Theorem 5 in Paixão et al., 2017).

$1 \leq k \leq d$ chosen later, this point is accepted, and from there the global optimum is reached before returning to the top of the cliff.

We estimate the probabilities for these events separately in order to get an overall lower bound on the probability of a trial being successful.

From any local peak there are $\binom{d}{k}$ search points at Hamming distance k that have $n-d+k$ ones. Considering only such mutations, the probability of a mutation increasing the number of ones from $n-d$ by k is at least

$$\begin{aligned}\mathrm{mut}(n-d, n-d+k) &\geq \frac{1}{n^k} \cdot \left(1 - \frac{1}{n}\right)^{n-1} \cdot \binom{d}{k} \\ &\geq \frac{1}{en^k} \cdot \left(\frac{d}{k}\right)^k.\end{aligned}$$

The probability of accepting such a move is

$$p_{\mathrm{fix}}(k-d+1/2) = \frac{e^{2\beta(d-k-1/2)}-1}{e^{2N\beta(d-k-1/2)}-1} \geq \frac{e^{2(d-k-1/2)}-1}{(9n)^{(d-k-1/2)}}.$$

We now fix $k := \lfloor d/e \rfloor$ and estimate the probability of making and accepting a jump of length k:

$$\begin{aligned}&\mathrm{mut}(n-d, n-d+k) \cdot p_{\mathrm{fix}}(k-d+1/2) \\ &\geq \frac{1}{en^k} \cdot \left(\frac{d}{k}\right)^k \cdot \frac{e^{2(d-k-1/2)}-1}{(9n)^{(d-k-1/2)}} \\ &= \Omega\left(n^{-d+1/2} \cdot \left(\frac{d}{k}\right)^k \cdot \left(\frac{e^2}{9}\right)^{d-k}\right) \\ &= \Omega\left(n^{-d+1/2} \cdot \left(e^{1/e} \cdot \left(\frac{e^2}{9}\right)^{1-1/e}\right)^d\right) \\ &= \Omega\left(n^{-d+1/2} \cdot \left(\frac{5}{4}\right)^d\right).\end{aligned}$$

Finally, we show that, if SSWM does make this accepted jump, with high probability it climbs up to the global optimum before returning to a search point in S_1. To this end we work towards applying the negative drift theorem to the number of ones in the interval $[a := \lceil n-d+k/2 \rceil, b := n-d+k]$ and show that, since we start in state b, a state a or less is unlikely to be reached in polynomial time.

We first show that the drift is typically equal to that on ONEMAX. For every search point with more than a ones, in order to reach S_1, at least $k/2$ bits have to flip. Until this happens, SSWM behaves like on ONEMAX and hence reaches either a global optimum or a point in S_1 in expected time $O(n \log n)$. The probability for a mutation flipping at least $k/2$ bits is at most $1/(k/2)! = (\log n)^{-\Omega(\log n)} = n^{-\Omega(\log\log n)}$, so the probability that this happens in expected time $O(n \log n)$ is still $n^{-\Omega(\log\log n)}$.

Assuming such jumps do not occur, we can then use drift bounds from the analysis of ONEMAX for states with at least a ones. However, we have to adapt the drift calculations from the proof of Theorem 5.1 to global mutations. Since the result and calculations are fairly similar, we include this translation in Lemma 6.11 in Appendix 6.C. Using Lemma 6.11 we know that the drift at x ones for $\beta = 1$ is at least

$$\Delta(x) \geq \Omega\left(\frac{n-x}{n}\right).$$

As in the proof of Theorem 5.2, we denote by $p_{x,j}$ the transition probability from a state with x ones to one with j ones. The probability of decreasing the current state is at most $p_{\mathrm{fix}}(-1) = O(1/n)$ due to Lemma 4.2. The probability of increasing the current state is at most $(n-x)/n$ as a necessary condition is that one out of $n-x$ zeros needs to flip. Hence for $x \leq b$, which implies $n-x = \omega(1)$, the self-loop probability is at least

$$p_{x,x} \geq 1 - O\left(\frac{1}{n}\right) - \frac{n-x}{n} = 1 - O\left(\frac{n-x}{n}\right).$$

Together, we get $\Delta(x) \geq \Omega(1-p_{x,x})$, establishing the first condition of the negative drift theorem (see Theorem 2.13).

Note that $p_{\mathrm{fix}}(1) = \frac{1-e^{-2}}{1-1/n} = \Omega(1)$, hence

$$1 - p_{x,x} \geq p_{x,x+1} \geq \frac{n-x}{en} \cdot p_{\mathrm{fix}}(1) = \Omega\left(\frac{n-x}{n}\right). \tag{6.9}$$

The second condition of the negative drift theorem follows for improving jumps from x to $x+j$, $j \geq 1$, from Lemma 2.1 and (6.9):

$$\begin{aligned} p_{x,x+j} &\leq \left(\frac{n-x}{n}\right)^j \cdot \frac{1}{j!} \cdot p_{\mathrm{fix}}(j) \\ &\leq \frac{n-x}{n} \cdot \frac{1}{j!} \\ &\leq (1-p_{x,x}) \cdot \frac{O(1)}{2^j}. \end{aligned}$$

For backward jumps we get, for $1 \le j \le k/2$, and n large enough,

$$\begin{aligned} p_{x,x-j} &\le p_{\text{fix}}(-j) \\ &\le \frac{e^{2j}}{e^{2Nj}-1} \\ &= \frac{e^{2j}}{(9n)^j - 1} \le 2^{-j}. \end{aligned}$$

Now the negative drift theorem can be applied with $r = O(1)$ and $\delta = 1$ and it yields that the probability of reaching a state of a or less in $n^{\omega(1)}$ steps is $n^{-\omega(1)}$.

This implies that following a length-k jump, a trial is successful with probability $1 - n^{-\omega(1)}$. This establishes $p_{\text{success}} := \Omega\left(n^{-d+1/2} \cdot \left(\frac{5}{4}\right)^d\right)$. Plugging this into (6.8), adding time R for the time to reach the top of the cliff initially, and using that $O(n^{1/2}\log n) \cdot (4/5)^d = e^{-\Omega(d)}$ for $d = \omega(\log n)$ yields the claimed bound. □

To complete the picture, we performed some experiments. As implied by Theorems 6.8 and 6.9, the following figures show how both the (1+1) EA and SSWM need superpolynomial time to optimise $\text{CLIFF}_{n/4}$ (Figure 6.8, left), however there is a superpolynomial gap between the runtimes of these algorithms (Figure 6.8, right).

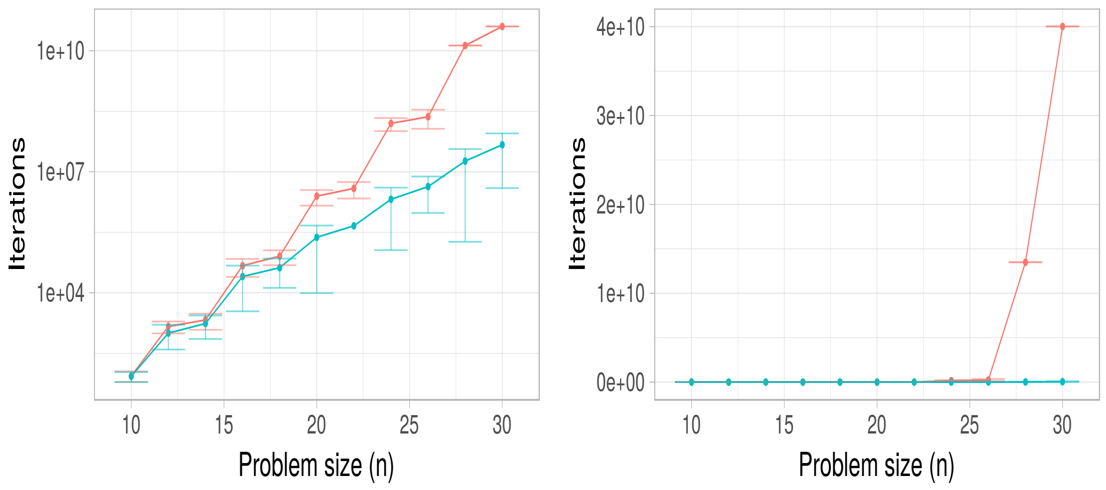


Fig. 6.8 Expected runtime of SSWM (blue) and the (1+1) EA (red) on $\text{CLIFF}_{n/4}$. Simulations were stopped after $n^d = n^{n/4}$ iterations (recall Theorem 6.8). SSWM's parameters were chosen according to Theorem 6.9 , i.e., $\beta = 1$ and $N\beta = 1/2\ln(9n)$. Results were averaged over 50 independent runs. For the left figure, a logarithmic scale with base 10 is used for the y-axis. Error bars include ± one standard deviation.

6.5 Conclusions

We presented an analysis of randomised search heuristics for crossing fitness valleys where no mutational bias exists and thus the probability for moving forwards or backwards on the path depends only on the fitness difference between neighbouring search points. Our focus was to highlight characteristics of valleys where an elitist selection strategy should be preferred to a non-elitist one and vice-versa. In particular, we compared the (1+1) EA using standard bit mutation with elitism against two algorithms using local mutations with non-elitism, namely SSWM and MA. To achieve our goals we presented a mathematical framework to allow the analysis of non-elitist algorithms on valleys and paths of concatenated valleys. We rigorously proved that while the (1+1) EA is efficient for valleys and valley paths up to moderate lengths, both SSWM and MA are efficient when the valleys and valley paths are not too deep.

Finally, we followed the natural research direction and let SSWM use global mutations. This allowed highlighting the benefits of combining both non-elitism and global mutations for overcoming local optima. We have shown that SSWM can effectively tunnel through sharp valleys such as posed by CLIFF_d, in contrast to elitist algorithms which get stuck for a long time on top of the cliff.

In addition, our results are meaningful for PG since crossing fitness valleys and highly rugged landscapes are very important topics in biology (Whitlock et al., 1995). As future work, we would like to check the agreement or disagreement of this chapter's results with Wright's shifting balance theory (Wright, 1932). This theory states that on fitness landscapes with multiple peaks where isolated populations might fail, parallel populations can cooperate to effectively adapt on such landscapes. We have identified families of fitness valleys that are intractable for SSWM, hence our results are in concordance with the first claim from Wright's theory (isolated populations fail). The remaining unanswered question from this chapter would be to analyse a distributed scenario where several SSWM algorithms cooperate.

Appendix 6.A Omitted Proofs from Subsection 6.2.2

Restating Lemma 6.4. *In the context of Lemma 6.3, properties (i) and (ii) imply that*

(i) $p_{\ell_1\to\ell_1-1} + p_{\ell_1\to\ell_1+1} = 1/c_1$ *for some constant* $c_1 \geq 1$

(ii) $1 - c_1 \cdot p_{\ell_1\to\ell_1+1} \cdot p^{\mathrm{GR}}_{\ell_1+1\to\ell_1} = 1/c_2$ *for some constant* $c_2 > 1$

(iii) $1 - c_1c_2 \cdot p_{\ell_1\to\ell_1-1} = 1/c_3$ *for some constant* $c_3 > 1$

Proof of Lemma 6.4. The first result follows directly from

$$\begin{aligned} &p_{\ell_1\to\ell_1+1},\ p_{\ell_1\to\ell_1-1} = \Omega(1) && \text{(property (i) of Lemma 6.3)} \\ &p_{\ell_1\to\ell_1-1} + p_{\ell_1\to\ell_1+1} \leq 1. \end{aligned}$$

For the second result, first we observe that $1 - c_1 \cdot p_{\ell_1\to\ell_1+1} \cdot p^{\mathrm{GR}}_{\ell_1+1\to\ell_1} < 1$. Then we need to prove a constant lower bound on $1 - c_1 \cdot p_{\ell_1\to\ell_1+1} \cdot p^{\mathrm{GR}}_{\ell_1+1\to\ell_1}$. For that, we start considering the term $p^{\mathrm{GR}}_{\ell_1+1\to\ell_1}$, which was defined in the framework's notation (see Definition 6.3). Using the results from the Gambler's Ruin problem (see Theorem 2.3) we can express $p^{\mathrm{GR}}_{\ell_1+1\to\ell_1}$ as

$$p^{\mathrm{GR}}_{\ell_1+1\to\ell_1} = \frac{1 - \left(\frac{p_{\ell_1\to\ell_1+1}}{p_{\ell_1+1\to\ell_1}}\right)^{\ell_2-1}}{1 - \left(\frac{p_{\ell_1\to\ell_1+1}}{p_{\ell_1+1\to\ell_1}}\right)^{\ell_2}} = \frac{p_{\ell_1+1\to\ell_1}}{p_{\ell_1\to\ell_1+1}} \cdot \frac{\left(\frac{p_{\ell_1\to\ell_1+1}}{p_{\ell_1+1\to\ell_1}}\right)^{\ell_2} - \frac{p_{\ell_1\to\ell_1+1}}{p_{\ell_1+1\to\ell_1}}}{\left(\frac{p_{\ell_1\to\ell_1+1}}{p_{\ell_1+1\to\ell_1}}\right)^{\ell_2} - 1} \leq \frac{p_{\ell_1+1\to\ell_1}}{p_{\ell_1\to\ell_1+1}} \tag{6.10}$$

where in the last step we have used that $p_{\ell_1+1\to\ell_1} < p_{\ell_1\to\ell_1+1}$ (property (ii) of Lemma 6.3) to upper bound the quotient by 1. Now we recover the second claim and introduce the already proven first result: $p_{\ell_1\to\ell_1-1} + p_{\ell_1\to\ell_1+1} = 1/c_1$ leading to

$$1 - c_1 \cdot p_{\ell_1\to\ell_1+1} \cdot p^{\mathrm{GR}}_{\ell_1+1\to\ell_1} = 1 - \frac{p_{\ell_1\to\ell_1+1} \cdot p^{\mathrm{GR}}_{\ell_1+1\to\ell_1}}{p_{\ell_1\to\ell_1-1} + p_{\ell_1\to\ell_1+1}}.$$

Using $p^{\mathrm{GR}}_{\ell_1+1\to\ell_1} \leq p_{\ell_1+1\to\ell_1}/p_{\ell_1\to\ell_1+1}$ obtained in Equation (6.10) yields

$$1 - c_1 \cdot p_{\ell_1\to\ell_1+1} \cdot p^{\mathrm{GR}}_{\ell_1+1\to\ell_1} \geq 1 - \frac{p_{\ell_1+1\to\ell_1}}{p_{\ell_1\to\ell_1+1} + p_{\ell_1\to\ell_1-1}} \geq \frac{p_{\ell_1\to\ell_1+1} - p_{\ell_1+1\to\ell_1}}{p_{\ell_1\to\ell_1+1} + p_{\ell_1\to\ell_1-1}} = \frac{1}{c}$$

for some constant $c > 1$. The last step follows since both numerator and denominator are constant due to properties (ii) and (i) from Lemma 6.3 respectively. The claimed equality from the statement with $1/c_2$ will hold for some constant c_2 in the range $c \geq c_2 > 1$.

Finally, for the third statement we start by observing that $1 - c_1c_2 \cdot p_{\ell_1\to\ell_1-1} < 1$. The remaining part of the proof is to show a constant lower bound on $1 - c_1c_2 \cdot p_{\ell_1\to\ell_1-1}$. For this, we introduce the already proven results for c_1 and c_2 obtaining

$$\begin{aligned}
&1 - c_1 c_2 \cdot p_{\ell_1 \to \ell_1 - 1} \\
&= 1 - \frac{c_1 \cdot p_{\ell_1 \to \ell_1 - 1}}{1 - c_1 \cdot p_{\ell_1 \to \ell_1 + 1} \cdot p^{\mathrm{GR}}_{\ell_1 + 1 \to \ell_1}} \\
&= 1 - \frac{p_{\ell_1 \to \ell_1 - 1}}{p_{\ell_1 \to \ell_1 - 1} + p_{\ell_1 \to \ell_1 + 1}} \cdot \frac{1}{1 - \frac{p_{\ell_1 \to \ell_1 + 1} \cdot p^{\mathrm{GR}}_{\ell_1 + 1 \to \ell_1}}{p_{\ell_1 \to \ell_1 - 1} + p_{\ell_1 \to \ell_1 + 1}}} \\
&= 1 - \frac{p_{\ell_1 \to \ell_1 - 1}}{p_{\ell_1 \to \ell_1 - 1} + p_{\ell_1 \to \ell_1 + 1} - p_{\ell_1 \to \ell_1 + 1} \cdot p^{\mathrm{GR}}_{\ell_1 + 1 \to \ell_1}} \\
&\geq 1 - \frac{p_{\ell_1 \to \ell_1 - 1}}{p_{\ell_1 \to \ell_1 - 1} + p_{\ell_1 \to \ell_1 + 1} - p_{\ell_1 + 1 \to \ell_1}} && \text{(Equation (6.10))} \\
&\geq 1 - \frac{p_{\ell_1 \to \ell_1 - 1}}{p_{\ell_1 \to \ell_1 - 1} + \varepsilon}, \ \ \varepsilon > 0 && \text{(property (ii) of Lemma 6.3)} \\
&= \frac{1}{c_3}, \ \ c_3 > 1. && \text{(property (i) of Lemma 6.3)}
\end{aligned}$$

□

Appendix 6.B Omitted Proofs from Subsection 6.2.4

Restating Lemma 6.7. *Consider a Gambler's Ruin problem as described in Theorem 2.3 with starting dollars $n_1 = 1$ and $n_2 = \ell - 1$. And probabilities p_1 and p_2 dependent on MA's acceptance function as follows*

$$p_1 = \frac{1}{2} \cdot e^{\alpha \Delta f} \qquad p_2 = \frac{1}{2}$$

where $\Delta f < 0$ and $\alpha |\Delta f| = \Omega(1)$. Then the winning probability of player one P_1 can be bounded as follows

$$\frac{-\alpha \Delta f}{e^{-\alpha \ell \Delta f}} < P_1^{\mathrm{GR-Met}} < \frac{e^{-\alpha \Delta f}}{e^{-\alpha \ell \Delta f} - 1}$$

and the expected duration of the game will be $E\left[T^{\mathrm{GR}}_{1,\ell}\right] = O(1)$.

Proof of Lemma 6.7. Since in general $1/2 + 1/2 \cdot e^{-\alpha |\Delta f|} < 1$ we have to make use of the Gambler's Ruin problem with self-loops (see Theorem 2.3). Let us start with the winning probabilities

$$P^{\mathrm{GR-Met}}_{1\to\ell_1} = \frac{1-\left(\frac{p_2}{p_1}\right)^{n_1}}{1-\left(\frac{p_2}{p_1}\right)^{n_1+n_2}} = \frac{1-e^{-\alpha n_1 \Delta f}}{1-e^{-\alpha(n_1+n_2)\Delta f}}.$$

For the upper bound:

$$P^{\mathrm{GR-Met}}_{1\to\ell_1} = \frac{1-e^{-\alpha\Delta f}}{1-e^{-\alpha\ell\Delta f}} = \frac{e^{-\alpha\Delta f}-1}{e^{-\alpha\ell\Delta f}-1} < \frac{e^{-\alpha\Delta f}}{e^{-\alpha\ell\Delta f}-1}.$$

For the lower bound:

$$P^{\mathrm{GR-Met}}_{1\to\ell_1} = \frac{e^{-\alpha\Delta f}-1}{e^{-\alpha\ell\Delta f}-1} > \frac{-\alpha\Delta f}{e^{-\alpha\ell\Delta f}-1} > \frac{-\alpha\Delta f}{e^{-\alpha\ell\Delta f}}.$$

Using again Theorem 2.3 for the expected duration of the game leads to

$$\mathrm{E}\left[T^{\mathrm{GR-Met}}_{1,\ell_1}\right] = \frac{n_1-(n_1+n_2)\cdot P^{\mathrm{GR-Met}}_{1}}{p_2^2-p_1^2} = \frac{1-\ell\cdot P^{\mathrm{GR-Met}}_{1}}{p_2^2-p_1^2} \leq \frac{1}{p_2^2\left(1-\frac{p_1^2}{p_2^2}\right)}$$

introducing p_1 and p_1 and noticing that $\Delta f < 0$ yields

$$\mathrm{E}\left[T^{\mathrm{GR-Met}}_{1,\ell_1}\right] \leq \frac{4}{1-e^{2\alpha\Delta f}} = \frac{4}{1-e^{-2\alpha|\Delta f|}} = O(1).$$

Where in the last step we have used that $\alpha|\Delta f| = \Omega(1)$ implies $e^{-2\alpha|\Delta f|} = 1-\Omega(1)$. □

Restating Theorem 6.4. *The expected number of function evaluations $E\left[T_f\right]$ for the Metropolis algorithm to reach $P_{\ell_1+\ell_2}$ from P_0 on* VALLEY *with $\ell_1,\ell_2 \in \mathbb{N}\setminus\{1\}$ and $d_1,d_2 \in R^+$ is*

$$E(T) = O\left(n\cdot e^{\alpha d_1(1+1/\ell_1)}\right) + \Theta(n\cdot\ell_2)$$
$$E(T) = \Omega\left(n\cdot e^{\alpha d_1(1-1/\ell_1)}\right) + \Theta(n\cdot\ell_2)$$

provided $\alpha d_1/\ell_1$, $\alpha d_2/\ell_2 = \Omega(1)$.

Proof of Theorem 6.4. Let's begin by checking the conditions of Lemma 6.5 for MA. First,*(i)* $p_{\ell_1\to\ell_1+1} = p_{\ell_1\to\ell_1-1} = 1/2$. Next, *(ii)* $p^2_{\ell_1\to\ell_1+1} = 1/4$ and $p_{\ell_1+1\to\ell_1} = e^{-\alpha d_2/\ell_2}/2$. This implies that condition (*ii*) only applies if $e^{-\alpha d_2/\ell_2} < \frac{1}{2} \Leftrightarrow \alpha d_2/\ell_2 > \ln 2$. Conditions (*iii*) and (*iv*) are valid since MA accepts solutions with a probability that depends only on Δf and is non-decreasing in Δf.The proof for the fourth condition follows directly from

$p_{\ell_1\to\ell_1+1}/p_{\ell_1+1\to\ell_1} = e^{\alpha d_2/\ell_2}$ and the condition $\alpha d_2/\ell_2 = \Omega(1)$. Since these conditions are satisfied, Lemmas 6.3 and 6.4 apply and the expected time is:

$$\mathrm{E}\left[T_{0\to\ell_1+\ell_2}\right] = \Theta\left(\frac{1}{p^{\mathrm{GR}}_{1\to\ell_1}}\cdot\left(\mathrm{E}\left[T^{\mathrm{GR}}_{1,\ell_1}\right]+\frac{p^{\mathrm{GR}}_{1\to0}}{p_{0\to1}}\right)\right).$$

For the upper bound, note that from Lemma 6.7:

$$\mathrm{E}\left[T^{\mathrm{GR-Met}}_{1,\ell_1}\right] = O(1)$$

as long as $\frac{\alpha d_1}{\ell_1} = \Omega(1)$. Then, using the bounds on P_1 from Lemma 6.7

$$\begin{aligned}
\mathrm{E}\left[T_{0\to\ell_1+\ell_2}\right] &= O\left(\frac{1}{p^{\mathrm{GR}}_{1\to\ell_1}}\cdot\left(\mathrm{E}\left[T^{\mathrm{GR-Met}}_{1,\ell_1}\right]+\frac{p^{\mathrm{GR}}_{1\to0}}{p_{0\to1}}\right)\right)\\
&= O\left(\frac{1}{p^{\mathrm{GR}}_{1\to\ell_1}}\cdot\left(O(1)+e^{\alpha d_1/\ell_1}\right)\right)\\
&= O\left(\frac{\ell_1}{\alpha d_1}e^{\alpha d_1}\cdot\left(O(1)+e^{\alpha d_1/\ell_1}\right)\right)\\
&= O\left(\frac{\ell_1}{\alpha d_1}e^{\alpha d_1(1+1/\ell_1)}\right)\\
&= O\left(e^{\alpha d_1(1+1/\ell_1)}\right).
\end{aligned}$$

For the lower bound, since $\mathrm{E}\left[T^{\mathrm{GR-Met}}_{1,\ell_1}\right] > 0$ and $\frac{p^{\mathrm{GR}}_{1\to0}}{p_{0\to1}} = e^{\alpha d_1/\ell_1} > 1$:

$$\begin{aligned}
\mathrm{E}\left[T_{0\to\ell_1+\ell_2}\right] &= \Omega\left(\frac{1}{p^{\mathrm{GR}}_{1\to\ell_1}}\cdot\left(\mathrm{E}\left[T^{\mathrm{GR-Met}}_{1,\ell_1}\right]+\frac{p^{\mathrm{GR}}_{1\to0}}{p_{0\to1}}\right)\right)\\
&= \Omega\left(\frac{1}{p^{\mathrm{GR}}_{1\to\ell_1}}\right) = \Omega\left(\frac{e^{\alpha d_1}-1}{e^{\alpha d_1/\ell_1}}\right)\\
&= \Omega\left(e^{\alpha d_1(1-1/\ell_1)}-e^{-\alpha d_1/\ell_1}\right)\\
&= \Omega\left(e^{\alpha d_1(1-1/\ell_1)}\right).
\end{aligned}$$

Finally, we add the $\Theta(\ell_2)$ term (Lemma 6.3) and multiply by the time needed for a relevant step $n/2$ (Lemma 6.2). □

Appendix 6.C Omitted Proofs from Section 6.4

Restating Lemma 6.10. *For all* $0 \le x < j \le n$,

$$\frac{\mathrm{mut}(x,j)}{\sum_{k=j}^{n} \mathrm{mut}(x,k)} \ge \frac{1}{2}.$$

Proof of Lemma 6.10. The proof consists of two parts:
1) The probability of improving by $j - x = k$ bits is at least twice as large as the probability of improving by $k+1$ bits, i.e. $\mathrm{mut}(x, x+k) \ge 2\mathrm{mut}(x, x+k+1)$ for any $0 \le x < j \le n$.
2) We use 1) to prove that $\frac{\mathrm{mut}(x,j)}{\sum_{m=j}^{n} \mathrm{mut}(x,m)} \ge \frac{1}{2}$.

Part 1) The probability to improve by k bits is

$$\mathrm{mut}(x, x+k) = \sum_{l=0}^{n} \binom{x}{l} \binom{n-x}{k+l} \left(\frac{1}{n}\right)^{k+2l} \left(1 - \frac{1}{n}\right)^{n-k-2l}$$

while the probability to improve by $k+1$ bits is

$$\mathrm{mut}(x, x+k+1) = \sum_{l=0}^{n} \binom{x}{l} \binom{n-x}{k+l+1} \left(\frac{1}{n}\right)^{k+2l+1} \left(1 - \frac{1}{n}\right)^{n-k-2l-1}.$$

We want to show that the following is true:

$$\begin{aligned}
&\mathrm{mut}(x, x+k) \ge 2\mathrm{mut}(x, x+k+1) \\
\Leftrightarrow &\sum_{l=0}^{n} \binom{x}{l} \binom{n-x}{k+l} \left(\frac{1}{n}\right)^{k+2l} \left(1 - \frac{1}{n}\right)^{n-k-2l} \ge 2\sum_{l=0}^{n} \binom{x}{l} \binom{n-x}{k+l+1} \left(\frac{1}{n}\right)^{k+2l+1} \left(1 - \frac{1}{n}\right)^{n-k-2l-1} \\
\Leftrightarrow &\sum_{l=0}^{n} \binom{x}{l} \binom{n-x}{k+l} (n-1)^{n-k-2l} \ge 2\sum_{l=0}^{n} \binom{x}{l} \binom{n-x}{k+l+1} (n-1)^{n-k-2l-1} \\
\Leftrightarrow &\sum_{l=0}^{n} \frac{x!(n-x)!}{l!(x-l)!} \frac{(n-1)^{n-k-2l}}{(n-x-k-l-1)!(k+l)!} \left[\frac{1}{(n-x-k-l)} - \frac{2}{(n-1)(k+l+1)}\right] \ge 0.
\end{aligned}$$

This holds if following holds for any $0 \le l \le n$

$$\begin{aligned}
&\left[\frac{1}{(n-x-k-l)} - \frac{2}{(n-1)(k+l+1)}\right] \ge 0 \\
\Leftrightarrow &(n-1)(k+l+1) \ge 2(n-x-k-l).
\end{aligned}$$

This is true for any $k \geq 1$ (thus for any $0 \leq x < j \leq n$).

Part 2) Using the above inequality $\text{mut}(x, x+k) \geq 2\text{mut}(x, x+k+1)$ we can bound every possible improvement better than k from above by

$$\text{mut}(x, x+k+l) \leq \left(\frac{1}{2}\right)^l \text{mut}(x, x+k)$$

for any $0 \leq l \leq n-x-k$. This can also be written as

$$\text{mut}(x, j+l) \leq \left(\frac{1}{2}\right)^l \text{mut}(x, j)$$

for any $0 \leq l \leq n-j$. This leads to

$$\begin{aligned}
\frac{\text{mut}(x,j)}{\sum_{m=j}^{n} \text{mut}(x,m)} &= \frac{\text{mut}(x,j)}{\sum_{l=0}^{n-j} \text{mut}(x, j+l)} \\
&\geq \frac{\text{mut}(x,j)}{\sum_{l=0}^{n-j} \left(\frac{1}{2}\right)^l \text{mut}(x,j)} \\
&= \frac{1}{\sum_{l=0}^{n-j} \left(\frac{1}{2}\right)^l} = \frac{1}{2 - \frac{1}{2^{n-j}}} \geq \frac{1}{2}
\end{aligned}$$

which proves Lemma 6.10. □

Lemma 6.11. *If $2(N-1)\beta \geq \ln(cn)$, for a constant $c > 1.2$ and $\beta \in \mathbb{R}^+$ then drift of SSWM with global mutations on* OneMax *can be bounded by*

$$\Delta(x) \geq p_{\text{fix}}(1) \cdot \frac{c(n-x) - 1.2}{cen}.$$

Proof. The drift (see Definition 2.12) can be expressed as a combination of a forward and a backward drift $\Delta(x) = \Delta^+(x) - |\Delta^-(x)|$. We start by computing a lower bound for the forward drift

$$\begin{aligned}
\Delta^+(x) &= \sum_{j=1}^{n-x} \text{mut}(x, x+j) \cdot j \cdot p_{\text{fix}}(j) \\
&\geq \text{mut}(x, x+1) \cdot p_{\text{fix}}(1) \\
&\geq \frac{n-x}{n} \left(1 - \frac{1}{n}\right)^{n-1} p_{\text{fix}}(1). \qquad (6.11)
\end{aligned}$$

Secondly we calculate the upper bound for the backward drift,

$$|\Delta^-(x)| = \sum_{j=1}^{x} \mathrm{mut}(x, x-j) \cdot j \cdot p_{\mathrm{fix}}(-j),$$

where j is now the number of new zeros. Bounding $\mathrm{mut}(x, x-j)$ from above by Lemma 2.1 and bounding $x/n \leq 1 - 1/n$ yields

$$\begin{aligned}|\Delta^-(x)| &\leq \sum_{j=1}^{x} \frac{1.14}{j!} \cdot \left(1-\frac{1}{n}\right)^{j} \cdot \left(1-\frac{1}{n}\right)^{n-j} \cdot j \cdot p_{\mathrm{fix}}(-j)\\ &\leq \sum_{j=1}^{x} \frac{1.14}{j!} \cdot \left(1-\frac{1}{n}\right)^{n-1} \cdot j \cdot p_{\mathrm{fix}}(-j).\end{aligned}$$

Separating the case $j = 1$ and bounding the remaining fixation probabilities by $p_{\mathrm{fix}}(-2)$

$$\begin{aligned}|\Delta^-(x)| &\leq 1.14\left(1-\frac{1}{n}\right)^{n-1} p_{\mathrm{fix}}(-1) + 1.14\left(1-\frac{1}{n}\right)^{n-1} \cdot p_{\mathrm{fix}}(-2) \cdot \sum_{j=2}^{x} \frac{1}{(j-1)!}\\ &\leq 1.14\left(1-\frac{1}{n}\right)^{n-1} \left(p_{\mathrm{fix}}(-1) + p_{\mathrm{fix}}(-2) \cdot e\right) \end{aligned} \tag{6.12}$$

where in the last step we have used $\sum_{j=2}^{x} \frac{1}{(j-1)!} \leq \sum_{j=1}^{\infty} \frac{1}{j!} = e - 1 \leq e$.

Let us now estimate an upper bound for $p_{\mathrm{fix}}(-2)$,

$$\begin{aligned}p_{\mathrm{fix}}(-2) &= \frac{e^{4\beta}-1}{e^{4N\beta}-1} = \frac{e^{2\beta}-1}{e^{2N\beta}-1} \cdot \frac{e^{2\beta}+1}{e^{2N\beta}+1}\\ &= p_{\mathrm{fix}}(-1) \cdot \frac{e^{2\beta}+1}{e^{2N\beta}+1} \leq p_{\mathrm{fix}}(-1) \cdot \frac{2e^{2\beta}}{e^{2N\beta}},\end{aligned}$$

using $2(N-1)\beta \geq \ln(cn)$ leads to

$$p_{\mathrm{fix}}(-2) \leq p_{\mathrm{fix}}(-1) \cdot \frac{2}{cn}. \tag{6.13}$$

Introducing (6.12) and (6.11) into $\Delta(x) = \Delta^+(x) - |\Delta^-(x)|$ leads to

$$\begin{aligned}\Delta(x) &\geq \left(1-\frac{1}{n}\right)^{n-1} \cdot \left(\frac{n-x}{n} \cdot p_{\mathrm{fix}}(1) - 1.14 p_{\mathrm{fix}}(-1) - 1.14e \cdot p_{\mathrm{fix}}(-2)\right)\\ &= \left(1-\frac{1}{n}\right)^{n-1} p_{\mathrm{fix}}(1) \cdot \left(\frac{n-x}{n} - 1.14\frac{p_{\mathrm{fix}}(-1)}{p_{\mathrm{fix}}(1)} - 1.14e\frac{p_{\mathrm{fix}}(-2)}{p_{\mathrm{fix}}(1)}\right).\end{aligned}$$

Using $p_{\text{fix}}(-2) \leq p_{\text{fix}}(-1) \cdot \frac{2}{cn}$ (Equation (6.13)) and then Lemma 4.3 leads to

$$\begin{aligned}
\Delta(x) &\geq \left(1 - \frac{1}{n}\right)^{n-1} p_{\text{fix}}(1) \cdot \left(\frac{n-x}{n} - 1.14\frac{p_{\text{fix}}(-1)}{p_{\text{fix}}(1)} - \frac{2.28e}{cn}\frac{p_{\text{fix}}(-1)}{p_{\text{fix}}(1)}\right) \\
&= \left(1 - \frac{1}{n}\right)^{n-1} p_{\text{fix}}(1) \cdot \left(\frac{n-x}{n} - \left(1.14 + \frac{2.28e}{cn}\right) \cdot e^{-2(N-1)\beta}\right).
\end{aligned}$$

Since $2(N-1)\beta \geq \ln(cn)$, $c > 1.2$ and assuming $n \geq 100$ (otherwise an $O(n)$ bound is trivial) we obtain

$$\Delta(x) \geq \left(1 - \frac{1}{n}\right)^{n-1} p_{\text{fix}}(1) \cdot \left(\frac{n-x}{n} - \frac{1.2}{cn}\right).$$

Using $c > 1.2$ we verify that $(n-x)/n - 1.2/(cn)$ is always positive for $x \geq 1$, hence we may bound $(1-1/n)^{n-1} \geq 1/e$, leading to

$$\begin{aligned}
\Delta(x) &\geq p_{\text{fix}}(1) \cdot \left(\frac{n-x}{en} - \frac{1.2}{cen}\right) \\
&\geq p_{\text{fix}}(1) \cdot \frac{c(n-x) - 1.2}{cen}.
\end{aligned}$$

□

Chapter 7

When is it Beneficial to Reject Improvements?

This chapter is based on the following publications:

1. Nallaperuma, S., Oliveto, P. S., Pérez Heredia, J., and Sudholt, D. (2017). When is it beneficial to reject improvements?. *Proceedings of the Genetic and Evolutionary Computation Conference 2017 (GECCO '17)*, pages 1455–1462. ACM.

2. Nallaperuma, S., Oliveto, P. S., Pérez Heredia, J., and Sudholt, D. (2017). On the Analysis of Trajectory-Based Search Algorithms: When is it Beneficial to Reject Improvements or to Exploit? Submitted to *Algorithmica*.

As seen in previous chapters, the SSWM algorithm belongs to the class of trajectory-based search heuristics which evolve a single lineage rather than using a population. Amongst single trajectory algorithms, well-known ones are (randomised) local search, simulated annealing, the Metropolis algorithm (MA) and simple classes of evolutionary algorithms such as the well-studied (1+1) EA and the (1+λ) EA. The main differences between SSWM and the (1+1) EA is that the latter only accepts new solutions if they are at least as good as the previous ones (a property called *elitism*), while SSWM can reject improvements and it may also accept non-improving solutions with some probability (known as *non-elitism*). This characteristic may allow SSWM to escape local optima by gradually descending the slope leading to the optimum rather than relying on large, but rare, mutations to a point of high fitness far away.

In Chapter 6, we rigorously analysed the performance of SSWM in comparison with the (1+1) EA for escaping local optima. The study from Sections 6.2 and 6.3 only allowed SSWM to use local mutations such that the algorithm had to rely exclusively on its non-elitism to escape local optima, hence to highlight the differences between elitist and non-elitist strategies. The analysis revealed that the expected time of the (1+1) EA to cross the valley (i.e., escape the local optimum) is exponential in the length of the valley (Theorem 6.1) while the expected time for SSWM can be exponential in the depth of the valley (Theorem 6.3).

However, other non-elitist trajectory-based algorithms such as the well-known Metropolis algorithm have the same asymptotic runtime as SSWM on fitness valleys (Theorem 6.4). While both algorithms rely on non-elitism to descend the valleys, it is not necessarily obvious that the algorithms should have the same runtime on the valleys, because they differ significantly in the probability of accepting improving solutions. In particular, MA always accepts improvements while SSWM may reject an improving solution with a probability that depends on the difference between the quality of the new and the previous solution (recall Figure 4.1).

The reader might notice that, we already showed in Section 5.4 that SSWM can profit from rejecting improvements to optimise BALANCE in polynomial time. However, we believe that the Metropolis algorithm would also be efficient on BALANCE. In this chapter, we investigate SSWM and MA with the goal of identifying function characteristics for which the two algorithms perform differently. Given that the main difference between the two is that SSWM may reject improvements, we aim to identify a class of functions where it is beneficial to do so and, as a result, identify an example where SSWM outperforms MA.

The roadmap is as follows. First, we show that our task is not trivial by proving that both algorithms converge to the same stationary distribution for equivalent parameters. While this result seems to have been known in evolutionary biology (see e.g. Sella and Hirsh, 2005) we are not aware of a previous proof in the literature. In Section 7.2 we define a simple fitness function (called *3 state model*) where two possible choices may be made from the initial point; one leading to a much larger fitness than the other. The idea is that, while MA should be indifferent to the choice, SSWM should pick one choice more often than the other. Although this intuition is true, it turns out that, due to MA' ability of escaping local optima, the mixing time for the 3 state model is small and afterwards the two algorithms behave equivalently as proven in the previous section. In Section 7.3 we extend the fitness function (leading to a *5 state model*) by adding two more states of extremely high fitness such that, once the algorithms have made their choice, the probability of escaping the local optima is very low. By tuning these high fitness points we can either reward or penalise a strategy that rejects small improvements. We capitalise on this by concatenating several 5 state models together (each of which we refer to as a *component*) and by defining a composite function that requires that a high number of correct choices are made by the algorithm. Then we show that for appropriate fitness values of the different states, SSWM achieves the target of the function and MA does not with overwhelming probability.

In Section 7.4 we consider other common single trajectory-based search algorithms to compare their performance on the identified function class with SSWM and MA. The reason that SSWM outperforms MA for the identified composite function is that the former algorithm tends to favour the acceptance of search points on the slope of largest uphill gradient while the latter algorithm accepts any improvement independent of its quality. Hence, we expect that also other algorithms that prefer improvements of higher quality over smaller ones (i.e., a characteristic often referred to as *exploitation*) to also perform well on the composite function. To this end we consider the well known Best-Improvement Local Search (BILS) algorithm that always selects the neighbouring point of highest fitness and compare it with a less exploitational local search strategy which accepts the first found improvement (FILS). Finally, we also consider a classical single trajectory evolutionary algorithm that favours exploitation. In order to achieve a fair performance comparison with SSWM and MA we consider the $(1,\lambda)$ RLS algorithm which, like the former algorithms, uses non-elitism and local mutations. The results show that BILS excels on the composite function while the $(1,\lambda)$ RLS needs a sufficiently large value of λ. Along the way we complement our theoretical findings with experiments which help to understand the complete picture.

7.1 A Common Stationary distribution

For a brief recall of the foundations of Markov chains and mixing times we refer the reader to Subsection 2.3.1. However for a detailed formal introduction we refer the interested reader to the documents by Aldous and Fill (2017), Jerrum and Sinclair (1996) or Levin et al. (2008).

We already showed in Theorem 2.2 that the stationary distribution $\pi(x)$ of MA assigns to each bitstring x a probability mass proportional to $e^{\alpha f(x)}$. The case for SSWM follows easily after noticing that for both algorithms $p_{\text{acc}}(\Delta f)/p_{\text{acc}}(-\Delta f) = e^{\gamma \Delta f}$, where $\gamma = \alpha$ for MA and $\gamma = 2\beta(N-1)$ for SSWM (Lemma 4.3). The next theorem uses this fact to show that both SSWM and MA have the same stationary distribution.

Theorem 7.1. *Consider SSWM (Algorithm 4.1) over a Markov chain with states $x \in \{0,1\}^n$ and a fitness function $f : \{0,1\}^n \to \mathbb{R}$. Then the stationary distribution of such process will be*

$$\pi(x) = \frac{e^{2(N-1)\beta f(x)}}{Z}$$

where $Z = \sum_{x \in \{0,1\}^n} e^{2(N-1)\beta f(x)}$.

Proof. The proof follows similar steps as the proof of Theorem 2.2. Let us start considering the left-hand side term from the detailed balance condition (2.6) to find out the stationary distribution.

$$\begin{aligned}\pi(x)\cdot p(x\to y) &= \frac{e^{2(N-1)\beta f(x)}}{Z}\cdot\frac{1}{n}\cdot p_{\mathrm{fix}}(f(y)-f(x))\\ &= \frac{e^{2(N-1)\beta f(x)}}{Z}\cdot\frac{1}{n}\cdot\frac{p_{\mathrm{fix}}(f(y)-f(x))}{p_{\mathrm{fix}}(f(x)-f(y))}\cdot p_{\mathrm{fix}}(f(x)-f(y)),\end{aligned}$$

since $p_{\mathrm{fix}}(\Delta f)/p_{\mathrm{fix}}(-\Delta f)=e^{2(N-1)\beta\Delta f}$ (Lemma 4.3) we obtain

$$\begin{aligned}\pi(x)\cdot p(x\to y) &= \frac{e^{2(N-1)\beta f(x)}}{Z}\cdot\frac{1}{n}\cdot e^{2(N-1)\beta(f(y)-f(x))}\cdot p_{\mathrm{fix}}(f(x)-f(y))\\ &= \frac{e^{2(N-1)\beta f(y)}}{Z}\cdot\frac{1}{n}\cdot p_{\mathrm{fix}}(f(x)-f(y))\\ &= \pi(y)\cdot p_{y\to x}.\end{aligned}$$ □

As we saw in Subsection 2.3.1, the distance between the current distribution and the stationary distribution is measured by the *total variation distance* (2.7). And we introduced the concept of *mixing time* (see Definition 2.11) as the time required for both distributions to be closer than a predefined threshold. After the mixing time, both algorithms will be close to the stationary distribution, hence any differing behaviour can only be shown before the mixing time. In the following, we aim to construct problems where the mixing time is large, such that SSWM and MA show different performance over a long period of time. In particular, we seek to identify a problem where the expected first hitting time of SSWM is less than the mixing time.

7.2 A 3 State Model

We first introduce a fitness function defined on 2 bits. We will analyse the behaviour of SSWM and MA on this function, before proceeding (in Section 7.3.1) to concatenate n copies of the fitness function to create a new function where SSWM drastically outperforms MA.

The idea is simple: we start in a search point of low fitness, and are faced with two improving moves, one with a higher fitness than the other. This construction requires 3 search points, which are embedded in a 2-dimensional hypercube as shown in Figure 7.1. The 4th possible bitstring will have a fitness of $-\infty$, making it inaccessible for both MA and SSWM. As common in evolutionary computation, we sometimes refer to the model states as *phenotypes* and their bitstring encoding as *genotypes*.

Considering the 3 relevant nodes of the Markov Chain, they form a valley structure tunable through two parameters a and b representing the fitness difference between the minimum and the local and global optimum respectively.

Definition 7.1 (3 state model)**.** *For any $b > a > 0$ and a bit-pair $\{0,1\}^2$ the 3 state model $f_3^{a,b}$ assigns fitness as follows:*

$$\begin{aligned} f_3^{a,b}(01) &= a, && \textit{(state 1)} \\ f_3^{a,b}(00) &= 0, && \textit{(state 2)} \\ f_3^{a,b}(10) &= b, && \textit{(state 3)} \end{aligned}$$

and $f_3^{a,b}(11) = -\infty$.

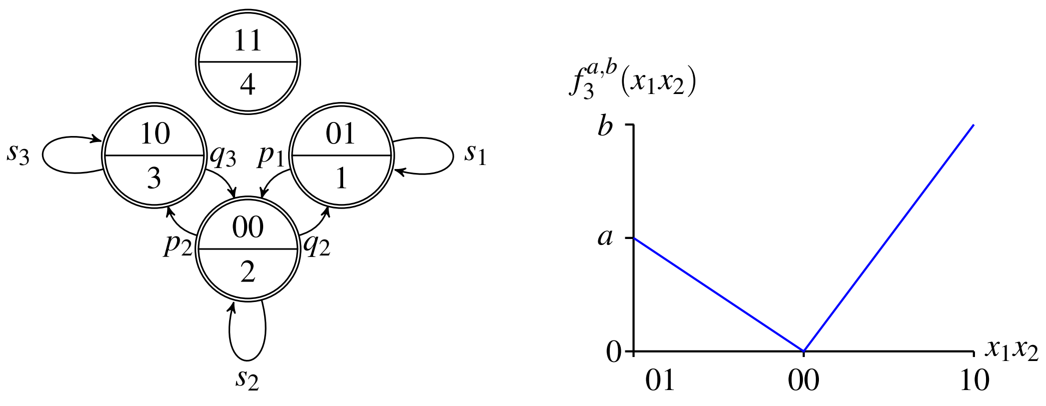


Fig. 7.1 Diagrams of the relevant nodes of $f_3^{a,b}(x_1x_2)$ at the genotype and phenotype level.

This model is loosely inspired by a two-locus (two bit) Dobzhansky-Muller incompatibility model (Orr, 1995; Unckless and Orr, 2009) in population genetics, where starting from an initial genotype (00 with fitness 0) there are two beneficial mutations (genotypes 01 with fitness $a > 0$ and 10 with fitness $b > 0$), but both mutations together are incompatible (genotype 11 with fitness $-\infty$).

This model is well suited for our purposes as MA is indifferent to the choice of the local optimum (fitness $a > 0$) and the global optimum (fitness $b > a$), hence it will make either choice from state 00 with probability $1/2$. SSWM, on the other hand, when parameterised accordingly, may reject a small improvement of fitness a more often than it would reject a larger improvement of $b > a$. Hence we expect SSWM to reach the global optimum with a probability larger than $1/2$ in just a relevant step (an iteration excluding self-loops). We make this rigorous in the following.

Since the analysis has similarities with the classical Gambler's Ruin problem (see Theorem 2.3) we introduce similar concepts to the ruin probability and the expected duration of the game.

Definition 7.2 (Notation)**.** *Consider a Markov Chain with only local probabilities.*

$$P(X_{t+1} = i \mid X_t = j) = \begin{cases} q_i & \text{if } j = i-1 \\ s_i = 1 - q_i - p_i & \text{if } j = i \\ p_i & \text{if } j = i+1 \\ 0 & \text{if } j \notin \{i-1, i, i+1\}. \end{cases}$$

We define absorbing probabilities ρ_i as the probability of hitting state k before state 1 *starting from i. Equivalently, we define expected absorbing times $E[T_{k\vee 1} \mid i]$ as the expected hitting time for either state* 1 *or k starting from i.*

Note that this definition may differ from the standard use of *absorbing* within Markovian processes. In our case the state k has an absorbing probability, but the state itself is not absorbing since the process may keep moving to other states.

The following lemma derives a closed form for the just defined absorbing probability, both for the general scheme 2.3 and for two specific algorithms. The obtained expression of $\rho_2 = p_2/(p_2+q_2)$ is simply the conditional probability of moving to the global optimum p_2 given that the process has moved, hence the factor $1-s_2$ in the denominator.

Theorem 7.2. *Consider any trajectory-based algorithm that fits in Algorithm 2.3 on $f_3^{a,b}$ starting from state* 2*. Then the absorbing probability of state* 3 *is*

$$\rho_2 = \frac{p_2}{p_2+q_2}.$$

And for the Metropolis algorithm and SSWM ($N \geq 2$) it is

$$\rho_2^{\text{MA}} = \frac{1}{2} \quad \rho_2^{\text{SSWM}} = \frac{p_{\text{fix}}(b)}{p_{\text{fix}}(b) + p_{\text{fix}}(a)} > \frac{1}{2}.$$

Proof. Let us start expressing the absorbing probability with a recurrence relation: $\rho_2 = p_2\rho_3 + q_2\rho_1 + (1-p_2-q_2)\rho_2$. Using the boundary conditions $\rho_3 = 1$ and $\rho_1 = 0$ we can solve the previous equation yielding $\rho_2 = p_2/(p_2+q_2)$.

The result for MA follows from introducing $p_2 = q_2$ since both probabilities lead to a fitness improvement. For SSWM the mutational component of p_2 and q_2 cancels out, yielding only the acceptance probabilities. Finally the lower bound of $1/2$ is due to state 3 having a fitness $b > a$. □

Note that SSWM's ability to reject improvements resembles a strategy of *best improvement* or *steepest ascent* (Smith, 2007): since the probability of accepting a large improvement is larger than the probability of accepting a small improvement, SSWM tends to favour the largest uphill gradient. MA, on the other hand, follows the first slope it finds, resembling a *first (or greedy) ascent* strategy.

However, despite these different behaviours, we know from Theorems 2.2 and 7.1 that both algorithms will eventually reach the same state. This seems surprising in the light of Theorem 7.2 where the probabilities of reaching the local versus global optimum from the minimum are potentially very different.

This seeming contradiction can be explained by the fact that MA is able to undo bad decisions by leaving the local optimum and going back to the starting point. Furthermore, leaving the local optimum has a much higher probability than leaving the global optimum. In the light of the previous discussion, MA' strategy in local optima resembles that of a *shallowest descent*: it tends to favour the smallest downhill gradient. This allows MA to also converge to the stationary distribution by leaving locally optimal states.

We show that the mixing time is asymptotically equal to the probability of accepting a move leaving the local optimum, state 1. Note that asymptotic notation is used with respect to said probability, as the problem size is fixed to 2 bits. To be able to bound the mixing time using Theorem 1.1 in Chen and Saloff-Coste (2013), we consider *lazy* versions of SSWM and MA: algorithms that with probability $1/2$ execute a step of SSWM or MA, respectively, and otherwise produce an idle step. This behaviour can also be achieved for the original algorithms by appending two irrelevant bits to the encoding of $f_3^{a,b}$.

Another assumption is that the algorithm parameters are chosen such that $\pi(3) \geq 1/2$. This is a natural assumption as state 3 has the highest fitness, and it is only violated in case the temperature is extremely high.

Theorem 7.3. *The mixing time of lazy SSWM and lazy MA on $f_3^{a,b}$ is $\Theta(1/p_{\text{acc}}(-a))$, provided $b > a > 0$ are chosen such that $\pi(3) \geq 1/2$.*

Proof. We use the transition probabilities from Figure 7.1. According to Theorem 1.1 in Chen and Saloff-Coste (2013), if $\pi(3) \geq 1/2$ then the mixing time of the lazy algorithms is of order $\Theta(t)$ where

$$t = \frac{1}{p_1} + \frac{\pi(1) + \pi(2)}{\pi(2) p_2}$$

As $p_1 = 1/2 \cdot p_{\text{acc}}(-a)$ this proves a lower bound $\Omega(1/p_{\text{acc}}(-a))$. For the upper bound, we bound t from above as follows, using $\pi(1)p_1 = \pi(2)q_2$ (the stationary distribution is reversible):

$$\begin{aligned} t &= \frac{1}{p_1} + \frac{\pi(1)+\pi(2)}{\pi(2)p_2} \\ &= \frac{1}{p_1} + \frac{\pi(1)}{\pi(2)p_2} + \frac{1}{p_2} \\ &= \frac{1}{p_1} + \frac{q_2}{p_2} \cdot \frac{1}{p_1} + \frac{1}{p_2} \leq \frac{3}{p_1} \end{aligned}$$

as $q_2/p_2 = p_{\mathrm{acc}}(a)/p_{\mathrm{acc}}(b) \leq 1$ and $p_2 \geq p_1$. Recalling that $p_1 = 1/2 \cdot p_{\mathrm{acc}}(-a)$ completes the proof. □

7.2.1 Experiments

We performed experiments to see the analysed dynamics more clearly. To this end, we considered a concatenated function

$$f(X) = \sum_{i=1}^{n} f_3^{M,a,b}(x_i)$$

consisting of n copies of the 3 state model (i.e. n components) x_i with $1 \leq i \leq n$, such that the concatenated function $f(x)$ returns the sum of the fitnesses of the individual components. Note that $2n$ bits are used in total. In our experiments, we chose $n = 100$ components.

In the case of SSWM we considered different population sizes $N = (10, 100)$ and scaling parameter values $\beta = (0.01, 0.1)$. For MA we choose a temperature of $1/\alpha$, such that $\alpha = 2(N-1)\beta$. This choice was made according to Theorems 2.2 and 7.1 such that both algorithms have the same stationary distribution. The algorithms are run for 10000 iterations. The fitness values for states representing local and global optimum are chosen as $a = 1$ and $b = 10$ respectively. We record the average and standard deviations of the number of components in the local and global optimum for 50 runs.

Figure 7.2 shows the number of components optimised (at both state 1 or state 3) for SSWM and MA. As suggested by Lemma 7.2, we observe on the left graph how SSWM (green curve) outperforms MA which only optimises correctly half of the components (purple curve). However, we know from Theorem 7.1 that both algorithms will eventually reach the same state. This is shown on the right plot of Figure 7.2 where the temperature was increased to facilitate the acceptance of worsening moves by MA.

The reason why the limit behaviour is only achieved on the right hand plot of Figure 7.2 is that the mixing time is inversely proportional to $p_{\mathrm{acc}}(-a)$ (Theorem 7.3), which in turn depends on a and the parameters of SSWM and MA. If the temperature is low (large α), the algorithms show a different behaviour before the mixing time, whereas if the temperature is high (small α), the algorithms quickly reach the same stationary distribution within the time budget given.

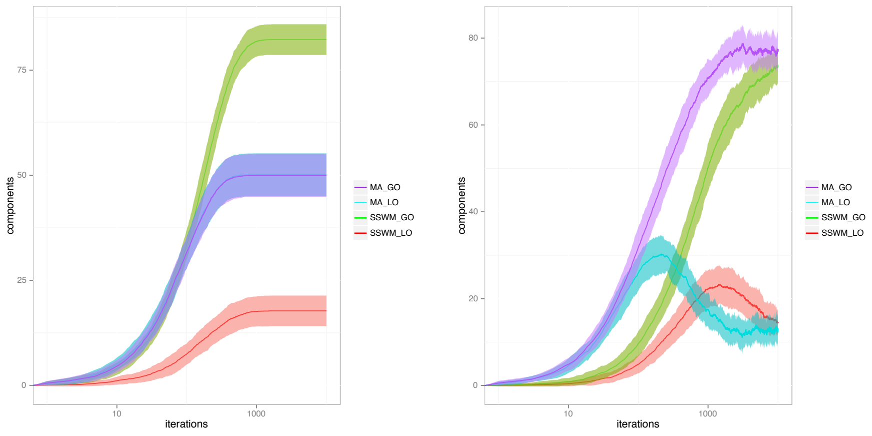


Fig. 7.2 Performance of SSWM with $N = 100$ and $\beta = 0.1$ (left) and $N = 10$ and $\beta = 0.01$ (right) on 100 concatenated components of the 3 state model. For MA the temperature was chosen such that $\alpha = 2(N-1)\beta$ in both cases. The average number of components ($\pm$ one standard deviation) in the global and local optimum are plotted for SSWM and for MA with colours red, green, purple and cyan respectively.

7.3 A 5 State Model

We saw in the previous section how two algorithms with different selection operators displayed the same limit behaviour. Moreover the mixing time was small for both algorithms despite the asymmetric valley structure of the function. This asymmetry favoured moving towards the steepest slope, a landscape feature from which SSWM benefits and MA is indifferent. However this feature also implied that it was easier climbing down from the shallowest slope, and MA successfully exploits this feature to recover from wrong decisions.

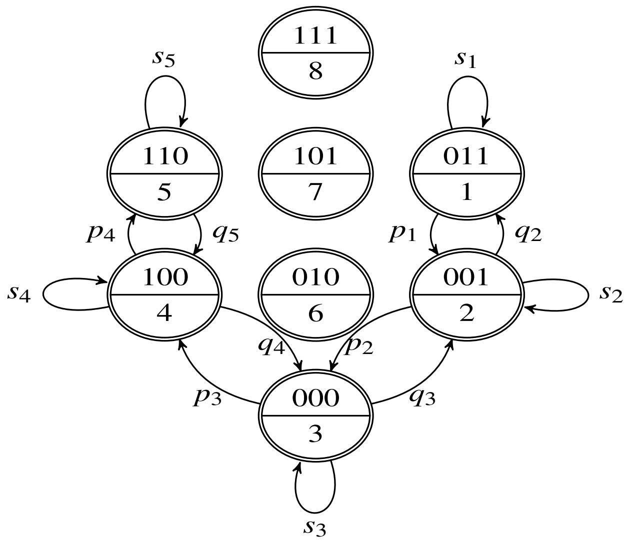
Making use of this results we build a new function where the previous local optimum will now be a transition point between the valley and the new local optimum. We will assign an extremely large fitness to this new search point. In this way we *lock in* bad decisions made by any of the two algorithms. In the same way, if the algorithm moves to the previous global optimum we offer a new search point with the highest fitness.

This new *5 state model* is shown in Figure 7.3, along with its encoding of genotypes in a 3-dimensional hypercube.

Definition 7.3 (5 state model)**.** *For any* $M' > M \gg b > a > 0$ *and a search point* $x \in \{0,1\}^3$ *the* 5 *state model* $f_5^{M,a,b,M'}$ *assigns fitness as follows*

$$\begin{aligned}
f_5^{M,a,b,M'}(011) &= M, && \textit{(state 1)}\\
f_5^{M,a,b,M'}(001) &= a, && \textit{(state 2)}\\
f_5^{M,a,b,M'}(000) &= 0, && \textit{(state 3)}\\
f_5^{M,a,b,M'}(100) &= b, && \textit{(state 4)}\\
f_5^{M,a,b,M'}(110) &= M' && \textit{(state 5)}
\end{aligned}$$

and $f_5^{M,a,b,M'}(010) = f_5^{M,a,b,M'}(101) = f_5^{M,a,b,M'}(111) = -\infty$.



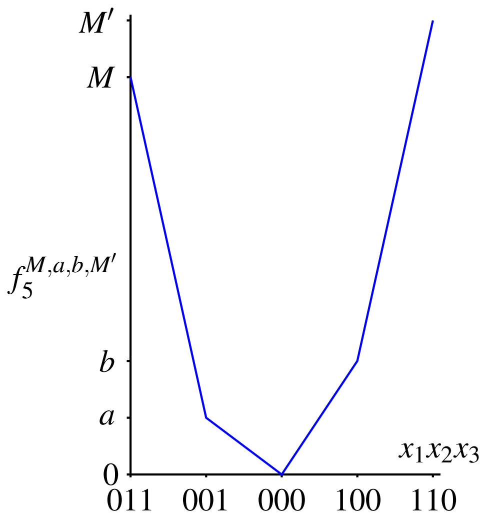


Fig. 7.3 Diagrams of the relevant nodes of $f_5^{M,a,b,M'}$ at the genotype and phenotype level.

Let us consider the Markov chain with respect to the above model. For simplicity we refer to states with the numbers 1-5 as in the above description.

Again, we will compute the absorbing probability for the global optimum (state 5 or 110 of the Markov Chain). Note that by choosing very large values of M and M', we can make the mixing time arbitrarily large, as then the expected time to leave state 1 or state 5 becomes very large, and so does the mixing time.

For simplicity we introduce the following conditional transition probabilities Q_i and P_i for each state i as

$$P_i := \frac{p_i}{p_i + q_i} \qquad Q_i := \frac{q_i}{p_i + q_i}. \tag{7.1}$$

By using this notation the following lemma derives a neat expression for the absorption probability $\rho_3 = P_3P_4/(Q_2Q_3 + P_3P_4)$. This formula can be understood in terms of events that can occur in 2 iterations starting from state 3. Since Q and P are conditioning on the absence of self-loops there will be only 4 events after 2 iterations, whose probabilities will be $\{Q_3Q_2, Q_3P_2, P_2Q_4, P_3P_4\}$. Therefore the expression $\rho_3 = P_3P_4/(Q_2Q_3 + P_3P_4)$ is just the success probability over the probability space.

Lemma 7.1. *Consider any trajectory-based algorithm that fits in Algorithm 2.3 on $f_5^{M,a,b,M'}$ starting from the node 3. Then the absorbing probability for state 5 is*

$$\rho_3 = \frac{P_3P_4}{Q_2Q_3 + P_3P_4}.$$

Proof. Firstly we compute the absorbing probabilities,

$$\begin{aligned}
&\rho_1 = 0\\
&\rho_2 = p_2\rho_3 + q_2\rho_1 + (1 - p_2 - q_2)\rho_2\\
&\rho_3 = p_3\rho_4 + q_3\rho_2 + (1 - p_3 - q_3)\rho_3\\
&\rho_4 = p_4\rho_5 + q_4\rho_3 + (1 - p_4 - q_4)\rho_4\\
&\rho_5 = 1
\end{aligned}$$

which can be rewritten using P_i and Q_i from Equation (7.1) and the two boundary conditions as

$$\begin{aligned}
&\rho_2 = P_2\rho_3\\
&\rho_3 = P_3\rho_4 + Q_3\rho_2\\
&\rho_4 = P_4 + Q_4\rho_3.
\end{aligned}$$

Solving the previous system for ρ_3 yields $\rho_3 = P_3 \cdot (P_4 + Q_4\rho_3) + Q_3P_2\rho_3$ which leads to

$$\rho_3 = \frac{P_3P_4}{1 - Q_3P_2 - P_3Q_4}.$$

Introducing $Q_3 = 1 - P_3$, $P_2 = 1 - Q_2$ and $Q_4 = 1 - P_4$ in the denominator yields the claimed statement. □

Now we apply the previous general result for the two studied heuristics. First, for the Metropolis algorithm one would expect the absorbing probability to be $1/2$ since it does not distinguish between improving moves of different magnitudes. However, it comes as a surprise that this probability will always be greater than $1/2$. The reason is again due to the fitness dependent acceptance probability of detrimental moves.

Theorem 7.4. *Consider MA starting from state* 3 *on* $f_5^{M,a,b,M'}$. *Then the absorbing probability for state* 5 *is*

$$\rho_3^{\text{MA}} = \frac{1+e^{-\alpha a}}{2+e^{-\alpha a}+e^{-\alpha b}} \geq \frac{1}{2}.$$

Proof. First let us compute the two conditional probabilities

$$Q_2 = \frac{1}{1+e^{-\alpha a}}, \quad P_4 = \frac{1}{1+e^{-\alpha b}}.$$

Now we invoke Lemma 7.1 but with $P_3 = Q_3 = 1/2$ since MA does not distinguish slope gradients. Hence,

$$\rho_3 = \frac{P_4}{Q_2+P_4} = \frac{1/\left(1+e^{-\alpha b}\right)}{1/(1+e^{-\alpha a})+1/\left(1+e^{-\alpha b}\right)} = \frac{1+e^{-\alpha a}}{2+e^{-\alpha a}+e^{-\alpha b}}.$$

Finally, using $\Delta f_3^2 \leq \Delta f_3^4$, it follows that $\rho_3^{\text{MA}} \geq 1/2$. □

Finally, for SSWM we were able to reduce the complexity of the absorbing probability to just the two intermediate points (states 2 and 4) between the valley (state 3) and the two optima (states 1 and 5). The obtained expression is reminiscent of the absorbing probability on the 3 State Model (Theorem 7.2). However, it is important to note that a and b were the fitness of the optima in $f_3^{a,b}$ and now they refer to the transition nodes between the valley and the optima.

Theorem 7.5. *Consider SSWM ($N \geq 2$) starting from state* 3 *on* $f_5^{M,a,b,M'}$. *Then the absorbing probability of state* 5 *is*

$$\rho_3^{\text{SSWM}} \geq \frac{p_{\text{fix}}(b)}{p_{\text{fix}}(b)+p_{\text{fix}}(a)} > \frac{1}{2}.$$

Proof. Let us start by computing the probabilities required by Lemma 7.1.

$$P_4 = \frac{1}{1+p_{\text{fix}}(-b)/p_{\text{fix}}(M'-b)} \qquad Q_2 = \frac{1}{1+p_{\text{fix}}(-a)/p_{\text{fix}}(M-a)}$$

$$P_3 = \frac{1}{1+p_{\text{fix}}(a)/p_{\text{fix}}(b)} \qquad Q_3 = \frac{1}{1+p_{\text{fix}}(b)/p_{\text{fix}}(a)}$$

Let us now focus on the term $Q_2Q_3/(P_3P4)$:

$$\frac{Q_2Q_3}{P_3P_4} = \frac{\left(1+\frac{p_{\text{fix}}(-b)}{p_{\text{fix}}(M'-b)}\right)}{\left(1+\frac{p_{\text{fix}}(-a)}{p_{\text{fix}}(M-a)}\right)} \cdot \frac{\left(1+\frac{p_{\text{fix}}(a)}{p_{\text{fix}}(b)}\right)}{\left(1+\frac{p_{\text{fix}}(b)}{p_{\text{fix}}(a)}\right)}$$

the last term is of the form $(1+x)/(1+1/x) = x$, hence it can be highly simplified to just $p_{\text{fix}}(a)/p_{\text{fix}}(b)$, yielding

$$\frac{Q_2Q_3}{P_3P_4} = \frac{\left(1+\frac{p_{\text{fix}}(-b)}{p_{\text{fix}}(M'-b)}\right)}{\left(1+\frac{p_{\text{fix}}(-a)}{p_{\text{fix}}(M-a)}\right)} \cdot \frac{p_{\text{fix}}(a)}{p_{\text{fix}}(b)}$$

since $0 < p_{\text{fix}}(-b) < p_{\text{fix}}(-a) < p_{\text{fix}}(M-a) < p_{\text{fix}}(M'-b) < 1$, we can bound $p_{\text{fix}}(-b)/p_{\text{fix}}(M'-b) \leq p_{\text{fix}}(-a)/p_{\text{fix}}(M-a)$ to obtain

$$\frac{Q_2Q_3}{P_3P_4} \leq \frac{\left(1+\frac{p_{\text{fix}}(-a)}{p_{\text{fix}}(M-a)}\right)}{\left(1+\frac{p_{\text{fix}}(-a)}{p_{\text{fix}}(M-a)}\right)} \cdot \frac{p_{\text{fix}}(a)}{p_{\text{fix}}(b)} = \frac{p_{\text{fix}}(a)}{p_{\text{fix}}(b)}.$$

Introducing this in Lemma 7.1 leads to

$$\rho_3 = \frac{1}{1+Q_2Q_3/(P_3P_4)} \geq \frac{1}{1+p_{\text{fix}}(a)/p_{\text{fix}}(b)} = \frac{p_{\text{fix}}(b)}{p_{\text{fix}}(b)+p_{\text{fix}}(a)}.$$

Finally, using $b > a$ we obtain the lower bound of $1/2$. □

7.3.1 An Example Where SSWM Outperforms the Metropolis algorithm

We now consider a smaller family of problems $f_5^{M,1,10,M'}$ and create an example where SSWM outperforms MA. In this simpler yet general scenario we can compute the optimal temperature for MA that will maximise the absorbing probability ρ_3^{MA}.

Lemma 7.2. *Consider the Metropolis algorithm on $f_5^{M,1,10,M'}$ starting from state 3. Then for any parameter $\alpha \in \mathbb{R}^+$ the absorbing probability ρ_3^{MA} of state 5 can be bounded as*

$$\rho_3^{\text{MA}}(\alpha) \leq \rho_3^{\text{MA}}(\alpha^*) < 0.63$$

where $\alpha^* = 0.312\ldots$ is the optimal value of α.

Proof. Introducing the problem settings ($a = 1$ and $b = 10$) in the absorbing probability from Theorem 7.4 yields

$$\rho_3^{\text{MA}}(\alpha) = \frac{1 + e^{-\alpha}}{2 + e^{-\alpha} + e^{-10\alpha}}$$

whose derivative is

$$\frac{d\rho_3^{\text{MA}}(\alpha)}{d\alpha} = \frac{e^{9\alpha\left(10e^{\alpha} - e^{10\alpha} + 9\right)}}{\left(e^{9\alpha} + 2e^{10\alpha} + 1\right)^2}.$$

By solving numerically this equation for $d(\rho_3^{\text{MA}}(\alpha))/d\alpha = 0$ with $\alpha > 0$ we obtain an optimal value of $\alpha^* = 0.312071\ldots$ which yields the maximum value of $\rho_3^{\text{MA}}(\alpha^*) = 0.623881\ldots$

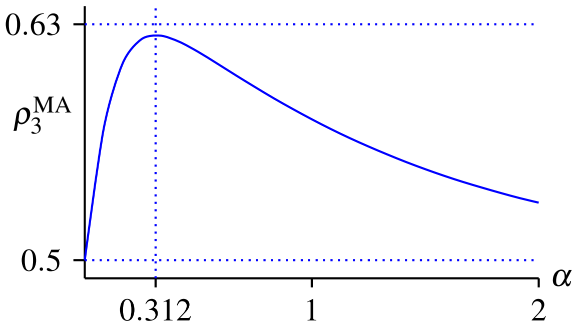


Fig. 7.4 Absorbing probability of MA on the 5-state model. □

Now that we have shown the optimal parameter for MA, we will find parameters such that SSWM outperforms MA. To obtain this we must make use of SSWM's ability of rejecting improvements. We wish to identify a parameter setting such that small improvements ($\Delta f = a = 1$) are accepted with small probabilities, while large improvements ($\Delta f = b = 10$) are accepted with a considerably higher probability. The following graph shows p_{fix} for different values of β. While for large β, $p_{\text{fix}}(1)$ and $p_{\text{fix}}(10)$ are similar, for smaller values of β there is a significant difference. Furthermore we can see that $p_{\text{fix}}(1) \leq 1/2$ i.e. the algorithm will prefer to stay put, rather than moving to the local optimum.

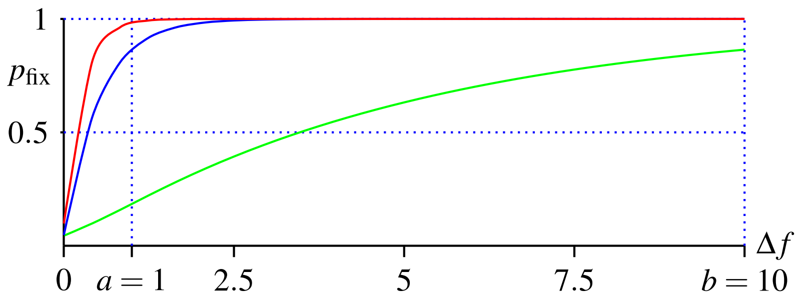


Fig. 7.5 Acceptance probability of SSWM with $N = 20$ and $\beta = (0.2, 2, 4)$ for the (green, blue, red) curves respectively.

In the following lemma we identify a range of parameters for which the desired effect occurs. The results hold for arbitrary population size, apart from the limit case $N = 1$ where SSWM becomes a pure random walk. The scaling factor β is the crucial parameter; only small values up to 0.33 will give a better performance than the Metropolis algorithm.

Lemma 7.3. *Consider SSWM on* $f_5^{M,1,10,M'}$ *starting from state* 3*. Then for* $\beta \in (0, 0.33]$ *and* $N \geq 2$ *the absorbing probability* ρ_3^{SSWM} *of state* 5 *is at least* 0.64*.*

Proof. Using the bound on ρ_3^{SSWM} from Theorem 7.5 with $a = 1$ and $b = 10$ we obtain

$$\rho_3^{\mathrm{SSWM}} \geq \frac{p_{\mathrm{fix}}(10)}{p_{\mathrm{fix}}(1) + p_{\mathrm{fix}}(10)} = \frac{1}{1 + p_{\mathrm{fix}}(1)/p_{\mathrm{fix}}(10)}.$$

We want to show that $\rho_3^{\mathrm{SSWM}} \geq 0.64$, which is equivalent to $p_{\mathrm{fix}}(1)/p_{\mathrm{fix}}(10) \leq 1/0.64 - 1 = 9/16$. Using the bounds $p_{\mathrm{fix}}(1) \leq 2\beta/(1 - e^{-2N\beta})$ and $p_{\mathrm{fix}}(10) \geq 20\beta/(1 + 20\beta)$ (see Lemma 4.2) we obtain

$$\frac{p_{\mathrm{fix}}(1)}{p_{\mathrm{fix}}(10)} \leq \frac{2\beta}{1 - e^{-2N\beta}} \cdot \frac{1 + 20\beta}{20\beta} = \frac{1 + 20\beta}{10\left(1 - e^{-2N\beta}\right)} \leq \frac{1 + 20\beta}{10\left(1 - e^{-4\beta}\right)}$$

where in the last step we have used $N \geq 2$. The obtained expression is always increasing with $\beta > 0$, hence we just need to find the value β^* for when it crosses our threshold value of $9/16$. Solving this numerically we found that the value is $\beta^* = 0.332423\ldots$, and the statement will be true for β values up to this cut off point. □

Now that we have derived parameter values for which SSWM has a higher absorbing probability on the 5 state model than MA for any temperature setting $1/\alpha$ (Lemma 7.2), we are ready to construct a function where SSWM considerably outperforms MA. We first define a concatenated function

$$f(X) = \sum_{i=1}^{n} f_5^{M,a,b,M'}(x_i) \tag{7.2}$$

consisting of n copies of the 5 state model (i.e. n components) x_i with $1 \leq i \leq n$, such that the concatenated function $f(x)$ returns the sum of the fitnesses of the individual components. Note that $3n$ bits are used in total. To ensure that the algorithms take long expected times to escape from each local optimum we set $M = n$ and $M' = 2n$ for each component x_i, apart from keeping $a = 1$ and $b = 10$, for which the absorbing probabilities from Lemmata 7.2 and 7.3 hold. Furthermore, we assume $2\beta(N-1) = \Omega(1)$ to ensure that SSWM remains in states 1 or 5 for a long time.

Theorem 7.6. *The expected time for SSWM and MA to reach either the local or global optimum of all the components of $f_5^{n,1,10,2n}$ is $O(n \log n)$. With overwhelming probability $1-e^{-\Omega(n)}$, SSWM with positive constant $\beta < 0.33$ and $N \geq 2$ has optimised correctly at least $(639/1000)n$ components while the Metropolis algorithm with optimal parameter $\alpha = 0.312\dots$ has optimised correctly at most $(631/1000)n$ components. The expected time for either algorithm to increase (or decrease) further the number of correctly optimised components by one is at least $e^{\Omega(n)}$.*

Proof. The expected time to reach either of the states 5 or 1 on the single-component 5 state model is a constant c for both algorithms. Hence, the first statement follows from an application of the coupon collector where each coupon has to be collected c times (Oliveto and Yao, 2011). The second statement follows by straightforward applications of Chernoff bounds (Lemma A.4) using that each component is independent and, pessimistically, that SSWM optimises each one correctly with probability $640/1000$ (i.e., Lemma 7.3) and MA with probability $630/1000$ (i.e., Lemma 7.2). The final statement follows because both algorithms with parameters $\Omega(1)$ accept a new solution, that is $\Omega(n)$ worse, only with exponentially small probability. □

The previous theorem ensures (with overwhelming probability) that after an expected polynomial time SSWM has correctly optimised at least $(8/1000)n$ more components than MA. Although this performance threshold is marginal, since the absorbing probabilities of SSWM and MA are both constants, with that of SSWM being higher than that of MA, we expect SSWM to achieve a higher fitness. We can amplify these potentially small differences by transforming our fitness function f with a step function $g(X)$ returning 1 if at least a certain number of components are optimised correctly (i.e. state 110 is found) and 0 otherwise:

$$g(X) := \begin{cases} 1 & \text{if at least } 0.635n \text{ components are in the global optimum state} \\ 0 & \text{otherwise.} \end{cases} \tag{7.3}$$

We use this to compose a function h where with overwhelming probability SSWM is efficient while MA is not:

$$h(X) = f(X) \cdot (1 - g(X)) + 2nM' \cdot g(X) \tag{7.4}$$

Note that $h(X) = f(X)$ while the step function $g(X)$ returns 0, and h attains a global optimum if and only if $g(X) = 1$. Our analysis transfers to the former case.

Corollary 7.1. *In the setting described in Theorem 7.6, with probability* $1-e^{-\Omega(n)}$ *SSWM finds an optimum on* $h(X)$ *after reaching either the local or global optimum on every component (which happens in expected time* $O(n\log n)$*), while MA requires* $e^{\Omega(n)}$ *steps with probability* $1-e^{-\Omega(n)}$.

Obviously, by swapping the values of M and M' in f, the function would change into one where preferring improvements of higher fitness is deceiving. As a result, SSWM would, with overwhelming probability, optimise at least 63.9% of the components incorrectly. However, although MA would optimise more components correctly than SSWM, it would still be inefficient on h.

7.3.2 Experiments

We performed experiments to study the performance of SSWM and MA on the 5 state model under several parameter settings. The experimental setting is similar to that of the 3 state model. We can see in Figure 7.6 how: while SSWM is able to reached the performance threshold imposed by $g(X)$, MA is not. As expected, both algorithms start with a g-value of 0 and hence they are optimising $f(X)$. However, for SSWM, once the dashed line on Figure 7.6 is reached, $g(X)$ drastically changes to 1 and $h(X)$ is optimised, hence the flat effect on the SSWM's curves.

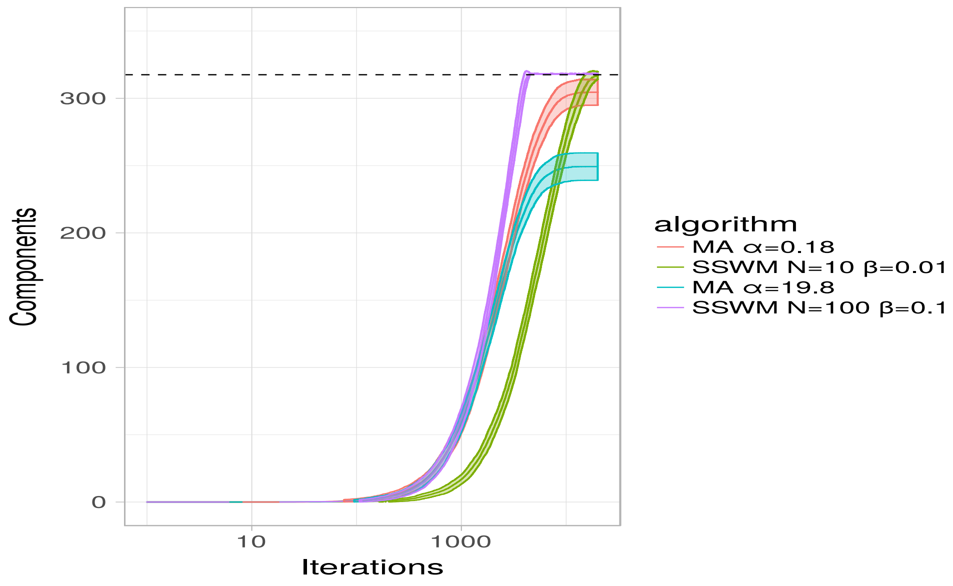


Fig. 7.6 Time evolution of the average number of components at state 5 by SSWM and MA when optimising $h(X)$ with 500 components of the 5 state model. Algorithm's parameters are in concordance with $\alpha = 2(N-1)\beta$. Results are averaged over 50 independent runs and the shadowed zones include $\pm$ one standard deviation. A logarithmic scale with base 10 is used for the x-axis. The dashed line ($y = 500 * 0.635$) indicates the threshold established on the definition of the step function $g(X)$.

We also plot the step function $g(X)$ as this is the most crucial term in $h(X)$. Again the results from Figure 7.7 are in concordance with the theory showing that SSWM outperforms MA. However, we observe that when choosing effective values of the temperature ($\alpha = 0.18$ in the figure) we can see that some runs of MA manages to optimise $g(X)$ yielding a non-zero expected value. The opposite effect can be seen for SSWM on the green curve, although its average g-value is much better than MA's, not all the runs made it to $g(X) = 1$. This is due to the chosen problem size not being enough large. If we recall Theorem 7.6, MA will in expectation optimise up to $(631/100)n$ components and SSWM at least $(639/1000)n$. This means that the gap for our chosen value of $n = 500$ is just 4 components, which can be achieved by some runs deviating from the expected behaviour. Due to limited computational resources we did not consider bigger values of n.

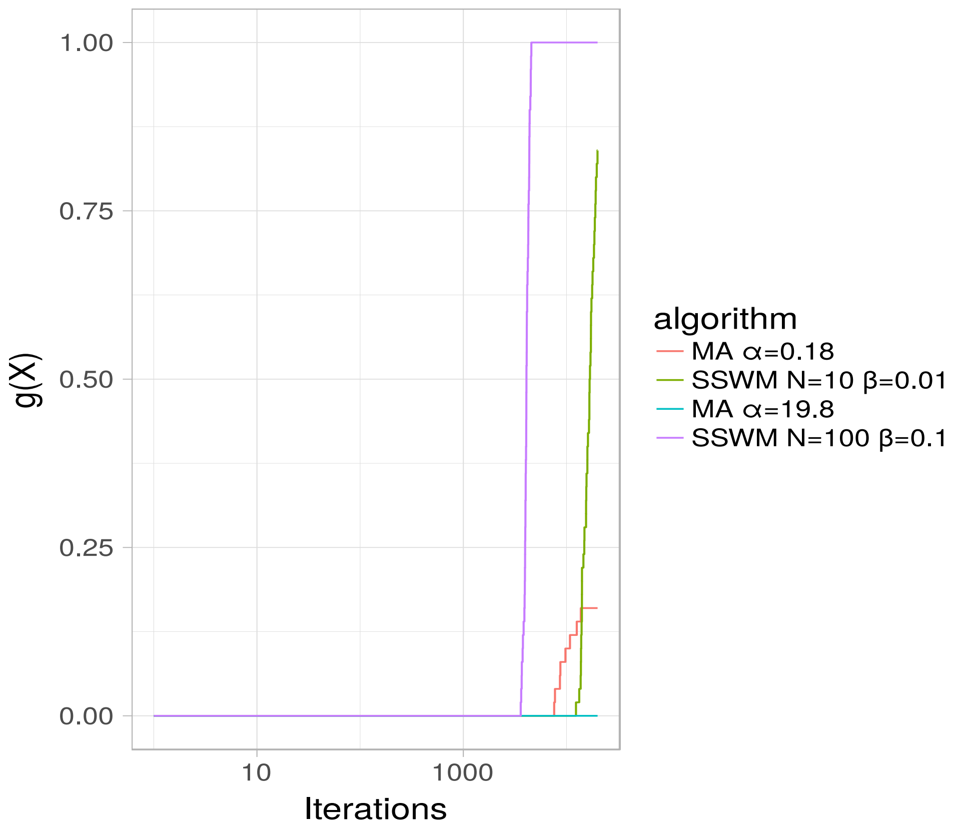


Fig. 7.7 Average $g(X)$ values over time for SSWM and MA when optimising $h(X)$ with 500 components of the 5 state model. Algorithm's parameters are in concordance with $\alpha = 2(N-1)\beta$. Results are averaged over 50 independent runs and a logarithmic scale with base 10 is used for the x-axis.

7.4 When is it Beneficial to Exploit?

We further analyse the performance of other common single trajectory-based search algorithms on the function classes we identified in the previous sections. The reason that SSWM outperforms MA for the identified composite function is that the former algorithm tends to favour the acceptance of search points on the slope of largest uphill gradient while the latter algorithm accepts any improvement independent of its quality. Hence, we expect that also other algorithms that prefer improvements of higher quality over smaller ones (i.e., a characteristic often referred to as *exploitation*) to also perform well on the composite function. A well known algorithm that predilects exploitation is the traditional local search strategy that selects the best improvement in the neighbourhood of the current search point (i.e., Best-Improvement Local Search). In particular, since a similar distinction between the behaviours of SSWM and MA is also present between BILS (Algorithm 2.10) and the local search strategy which selects the first found improvement (Algorithm 2.9) in the current neighbourhood, we will analyse the performance of these two algorithms. This also relates to previous work where the choice of the pivot rule was investigated in local search and memetic algorithms that combine evolutionary algorithms with local search (Gießen, 2013; Sudholt, 2011a; Wei and Dinneen, 2014).

FILS and BILS, like any Algorithm 2.2 with local mutations, can only explore the Hamming neighbourhood in one iteration. FILS will keep producing Hamming neighbours until it finds an improvement, whilst BILS computes the set of all neighbours and chooses one of those with the highest fitness. Both algorithms stop when there is no improving neighbour.

We will also consider a classical single trajectory evolutionary algorithm that favours exploitation. In order to achieve a fair performance comparison with SSWM and MA we consider the (1,λ) RLS algorithm which, like the former algorithms, uses non-elitism and local mutations. The algorithm creates λ new solutions, called *offspring*, at each step by mutating the current search point, and then it selects the best offspring, independent of whether it is an improvement. If the number of offspring λ is sufficiently large, then with high probability the slope with steepest gradient will be identified on one component. Since the analysis of the algorithm on the concatenated function is hard we rely on empirical tests to evaluate its performance.

The pseudo-code of the (1,λ) RLS was given in Algorithm 2.8 on page 22. This optimiser produces λ offspring by flipping one bit chosen uniformly at random independently for each offspring, and then choosing a best one to survive to the next generation. Although the selection mechanism picks the best offspring for survival, the (1,λ) RLS is not an elitist algorithm. Since the parent genotype is left out the fitness comparison, if the λ children have a lower fitness than the current solution, then the algorithm will move to a search point of lower fitness (even if the fitness is $-\infty$).

7.4.1 Analysis for the 3 State Model

We first derive the absorbing probabilities of the three new algorithms (FILS, BILS and the (1,λ) RLS) on the 3 state model. Theorem 7.7 confirms that BILS optimises the 2-bit function with probability 1, while FILS only does so with probability $1/2$. On the other hand, Theorem 7.8 reveals that the (1,λ) RLS always outperforms FILS for any $\lambda > 1$ and converges to the performance of BILS as the offspring population size λ increases.

Theorem 7.7. *Consider FILS and BILS on $f_3^{a,b}$ starting from state 2. Then the absorbing probabilities of state 3, respectively, are*

$$\rho_2^{\text{FILS}} = \frac{1}{2} \qquad \textit{and} \qquad \rho_2^{\text{BILS}} = 1.$$

Proof. FILS will produce either state 1 or state 3 (both with probability $1/2$) and accept the fitness change. Hence, like MA, FILS has transition probabilities $p_2 = q_2$ which, after a direct application of Theorem 7.2, yields the claimed result. On the other hand, BILS will produce both state 1 and state 3, and move to the latter since it has higher fitness. Hence, $q_2 = 0$ and $p_2 = 1$ which leads to an absorbing probability of 1 by Theorem 7.2. □

Theorem 7.8. *Consider the (1,λ) RLS on $f_3^{a,b}$ starting from state 2. Then, the absorbing probability of state 3 is*

$$\rho_2^{\textit{(1,}\lambda\textit{) RLS}} \geq 1 - 2^{-\lambda}.$$

Proof. In order for the (1,λ) RLS to move to state 3 from state 2 it suffices to create just one offspring at state 3 (the global optimum). The probability of creating such a search point is just the probability of choosing the first bit to be flipped, which is $1/2$. Then, with probability $(1 - 1/2)^\lambda = 2^{-\lambda}$ none of the λ offspring will be at state 3. And, the probability of at least one child being at the global optimum is $1 - 2^{-\lambda}$.

Hence, $p_2 = 1 - 2^{-\lambda}$ and since every mutation of state 2 leads to either state 1 or state 3, $q_2 = 1 - p_2 = 2^{-\lambda}$. Introducing this in Theorem 7.2 we obtain $\rho_2 = p_2$. □

7.4.2 Analysis for the 5 State Model

We now derive the absorbing probabilities of the three algorithms for the 5 state model. The absorbing probabilities for BILS and FILS as stated in the theorem below are the same as for the 3 state model.

Theorem 7.9. *Consider FILS and BILS on $f_5^{M,a,b,M'}$ starting from state 3. Then the absorbing probabilities of state 5, respectively, are*

$$\rho_3^{\mathrm{FILS}} = \frac{1}{2} \qquad \textit{and} \qquad \rho_3^{\mathrm{BILS}} = 1.$$

Proof. For FILS, a direct application of Lemma 7.1 with $P_4 = 1$, $P_3 = 1/2$, $Q_2 = 1$ and $Q_3 = 1/2$ yields an absorbing probability of $1/2$. For BILS, Lemma 7.1 with $P_4 = 1$, $P_3 = 1$, $Q_2 = 1$ and $Q_3 = 0$ yields an absorbing probability of 1. □

Interestingly, the analysis of (1,λ) RLS on the 5 state model turns out to be more complex than that of SSWM, MA, and (1,λ) RLS on the 3 state model as for the 5 state model it is possible for the algorithm to reach search points of fitness $-\infty$. This is because the non-absorbing states have Hamming neighbours of fitness $-\infty$, and such a search point is reached in case all λ offspring happen to have this fitness. While the genotypic encoding was irrelevant in all previous settings, it does become relevant in the following analysis.

The following Theorem 7.10 shows that the absorbing probability of the (1,λ) RLS converges to 1 slightly more slowly as λ increases than the one derived for the 3 state model.

Theorem 7.10. *Consider the (1,λ) RLS starting from state 3 on $f_5^{M,a,b,M'}$. Then the absorbing probability of state 5 is*

$$\rho_3^{(1,\lambda)\,RLS} = \frac{1-(2/3)^{\lambda}}{1-(1/3)^{\lambda}}.$$

Proof. Since the (1,λ) RLS can move to states with a fitness of $-\infty$, the diagram from Figure 7.3 is incomplete. However, let us focus now on the Hamming neighbours of each state. Recall that our genotype encoding of the 5 state model is based on 3 bits. We observe that, apart from the two maximal states (states 1 and 5), the three neighbours of each state have mutually different fitness values. Hence, we denote by p, q and r the transition probabilities towards the neighbour with the highest, intermediate and lowest fitness, respectively. Using this notation, we can express the absorbing probabilities as

$$\begin{aligned}
\rho_1 &= 0\\
\rho_2 &= q\rho_3 + r\rho_7\\
\rho_3 &= q\rho_2 + p\rho_4 + r\rho_6\\
\rho_4 &= q\rho_3 + p + r\rho_7\\
\rho_5 &= 1\\
\rho_6 &= r\rho_3 + p\\
\rho_7 &= q\rho_2 + p\rho_4 + r\rho_8\\
\rho_8 &= p + r\rho_7.
\end{aligned}$$

We now move to a matrix formulation of the form $A\rho = b$. But first, we introduce ρ_8 in ρ_7 and we no longer consider the trivial $\rho_1 = 0$ and $\rho_5 = 1$, hence $\rho = (\rho_2, \rho_3, \rho_4, \rho_6, \rho_7)^\top$, leading to

$$\begin{pmatrix} 1 & -q & 0 & 0 & p+q-1 \\ -q & 1 & -p & p+q-1 & 0 \\ 0 & -q & 1 & 0 & p+q-1 \\ 0 & p+q-1 & 0 & 1 & 0 \\ -q & 0 & -p & 0 & 1-(1-p-q)^2 \end{pmatrix} \cdot \begin{pmatrix} \rho_2 \\ \rho_3 \\ \rho_4 \\ \rho_6 \\ \rho_7 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ p \\ p \\ (1-p-q)p \end{pmatrix}.$$

The solution will be $\rho = A^{-1}b$, but we are just interested in ρ_3. Then, taking the second row of A^{-1} (here denoted as A_2^{-1}) we can express the absorbing probability as $\rho_3 = A_2^{-1}b^\top$. By standard matrix calculations, we obtain

$$A_2^{-1} = \frac{1}{(q-1)(p+q)} \cdot \left(-q, \ \tfrac{1}{p+q-2}, \ -p, \ \tfrac{-p-q+1}{p+q-2}, \ \tfrac{-p-q+1}{p+q-2} \right),$$

which can be verified with the expression $A^\top \left(A_2^{-1}\right)^\top = (0,1,0,0,0)$. Finally, we compute $\rho_3 = A_2^{-1}b^\top$ as follows:

$$\rho_3 = \frac{1}{(q-1)(p+q)} \cdot \left(-q, \ \tfrac{1}{p+q-2}, \ -p, \ \tfrac{-p-q+1}{p+q-2}, \ \tfrac{-p-q+1}{p+q-2} \right) \cdot \begin{pmatrix} 0 \\ 0 \\ p \\ p \\ (1-p-q)p \end{pmatrix}.$$

Which highly simplifies with the following calculations.

$$\begin{aligned}
\rho_3 &= \frac{-p^2+p(-p-q+1)/(p+q-2)+p(-p-q+1)^2/(p+q-2)}{(q-1)(p+q)} \\
&= \frac{p}{p+q}\cdot\frac{-p(p+q-2)+(-p-q+1)+(-p-q+1)^2}{(q-1)(p+q-2)} \\
&= \frac{p}{p+q}\cdot\left(\frac{-p}{q-1}+\frac{(-p-q+1)(2-p-q)}{(q-1)(p+q-2)}\right) \\
&= \frac{p}{p+q}\cdot\left(\frac{-p}{q-1}+\frac{p+q-1}{q-1}\right) \\
&= \frac{p}{p+q}. \qquad (7.5)
\end{aligned}$$

Finally, we just have to introduce the values of p and q. First, to move to the neighbour with the highest fitness, it is sufficient to produce one offspring at the desired search point. Noticing that $(1-1/3)^\lambda$ is the probability that none of the offspring are at the best neighbour, it follows that $p = 1-(1-1/3)^\lambda = 1-(2/3)^\lambda$. In order to move to the neighbour with the lowest fitness, all λ offspring must be equal to said neighbour, which happens with probability $r = (1/3)^\lambda$. Finally, $q = 1-p-r = (2/3)^\lambda - (1/3)^\lambda$. Introducing these values in Equation (7.5) leads to the claimed statement. □

Introducing $\lambda \geq 3$ in the expression obtained in Theorem 7.10, which is monotonically non-decreasing with λ, leads to

$$\rho_3^{(1,\lambda)\,\mathrm{RLS}} \geq \frac{1-(2/3)^3}{1-(1/3)^3} = \frac{1-8/27}{1-1/27} = \frac{19}{26} = 0.7307\ldots \geq 0.64.$$

Hence already an offspring population size of $\lambda = 3$ is sufficient to raise the success probability above that of the Metropolis algorithm with optimal parameters.

However, it is not straightforward to translate our results from one component $f_5^{M,a,b,M'}$ to n components. Unlike for SSWM and MA, on $n \gg 1$ components the (1,λ) RLS is likely to perform mutations in different components. Our analysis from Theorem 7.10 breaks down as all transition probabilities rely on the fact that all λ mutations concern the same component.

The dynamics on $n \gg 1$ components seem very different to the dynamics on one component, and quite complex. We therefore resort to experiments to shed light on the performance of (1,λ) RLS on n components and our composite function h.

7.4.3 Experiments

We present experimental results to understand the dynamics of the (1,λ) RLS on concatenated components of the 5 state model. Figure 7.8, shows the behaviour of the (1,λ) RLS when optimising $f(X)$ with 100 components. It is important to note that this setting does not exactly match the one from Figure 7.6, as there it was $h(X)$ the function that was being optimised. The only difference, is that in Figure 7.8 the algorithms can keep optimising components once the dashed line ($g(X) = 1$) is reached.

We observe an interesting effect for small values of λ, the algorithm starts accumulating components at state 5, however, at some point in time, the fitness decreases to that of a random configuration. This is due to the fact that states 6, 7 and 8 have a value of $-\infty$ for $f_5^{M,a,b,M'}$. If at some point in time, the algorithm sets just one component to either of these states, the total fitness $f(X)$ will be $-\infty$, no matter the fitness of the remaining components. Then, all that the (1,λ) RLS sees are points of equal fitness and it just chooses one uniformly at random. Obviously, the bigger the λ, the smaller the probability of sampling a point with $f(X) = -\infty$ in the first place and therefore, as seen in the figure, big values of λ manage to reach the threshold imposed by $g(X)$.

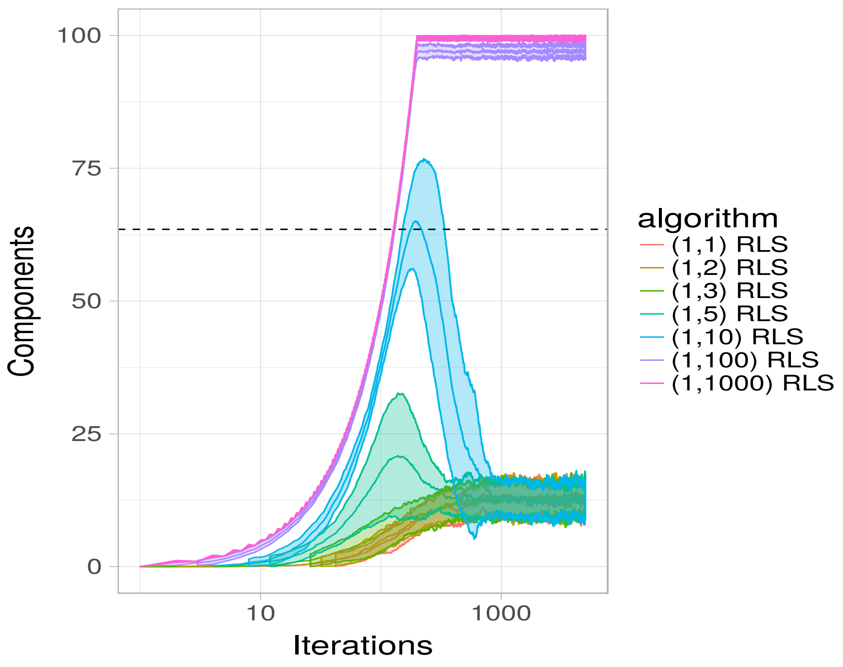


Fig. 7.8 Time evolution of the average number of components correctly optimised by the $(1,\lambda)$ RLS on 100 concatenated components of the 5 state model. Results are averaged over 50 independent runs and the shadowed zones include $\pm$ one standard deviation. A logarithmic scale with base 10 is used for the x-axis. The dashed line ($y = 63.5$) indicates the threshold established on the definition of the step function $g(X)$.

We now move to the study of the (1,λ) RLS when optimising $h(X)$. This is shown in Figure 7.9 by plotting the step function $g(X)$ as this is the most crucial term in $h(X)$. As suggested by Figure 7.8, it is needed a sufficiently large value of λ to ensure that all runs optimise $g(x)$ and by extension $h(X)$.

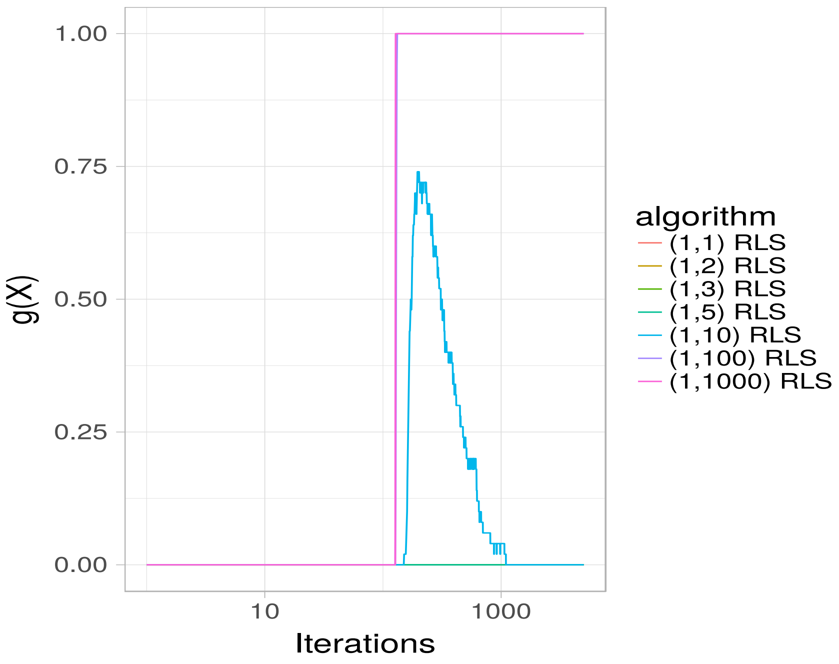


Fig. 7.9 Average $g(X)$ values over time for the $(1,\lambda)$ RLS when optimising $h(X)$ for 100 components of the 5 state model. Algorithm's parameters are in concordance with $\alpha = 2(N-1)\beta$. Results are averaged over 50 independent runs and a logarithmic scale with base 10 is used for the x-axis.

We conclude the subsection by presenting in Figure 7.10 a comparison graph that plots the performance of all the algorithms considered in this chapter. While BILS optimises all the components, the performance of SSWM and the $(1,\lambda)$ RLS is comparable and outperform the other algorithms. In particular, they both identify correctly a sufficient number of components such that they find the optimum of the composite function h.

7.5 Conclusions

We have presented a rigorous comparison of the non-elitist SSWM and Metropolis algorithms. Their main difference is that SSWM may reject improving solutions while MA always accepts them. Nevertheless, we prove that both algorithms have the same stationary distribution, and

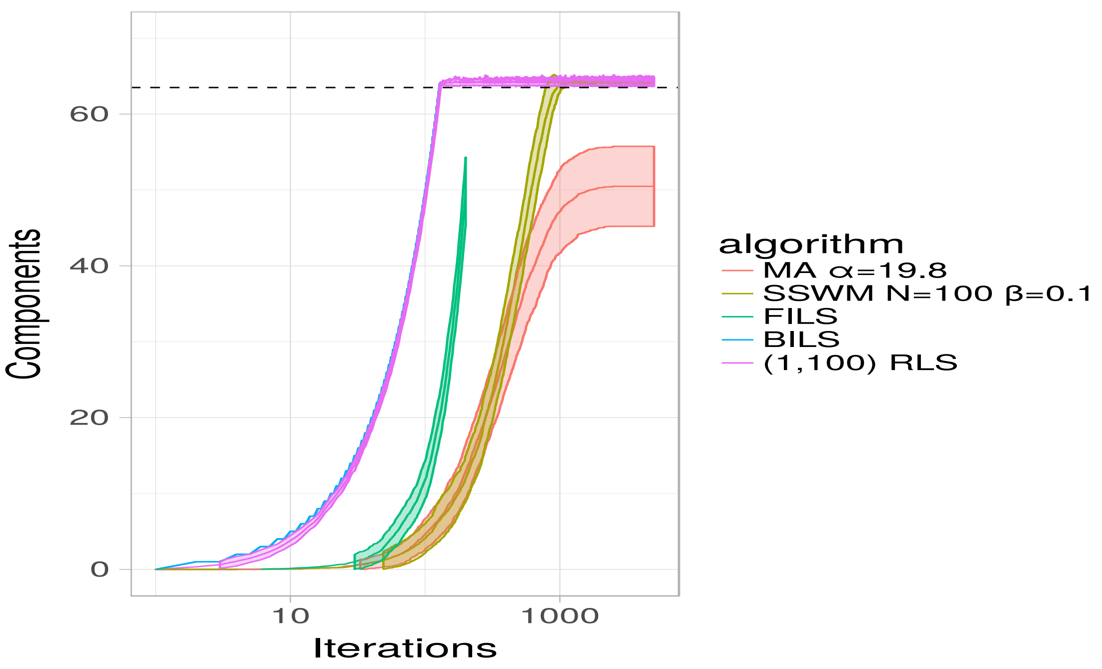


Fig. 7.10 Time evolution of the average number of components correctly optimised by all the algorithms when optimising $h(X)$ with 100 concatenated components of the 5 state model. Results are averaged over 50 independent runs and the shadowed zone includes ± one standard deviation. A logarithmic scale with base 10 is used for the x-axis. The dashed line ($y = 63.5$) indicates the threshold established on the definition of the step function $g(X)$.

they may only have considerably different performance on optimisation functions where the mixing time is large.

Our analysis on a 3 state model highlights that a simple function with a local optimum of low fitness and a global optimum of high fitness does not allow the required large mixing times. The reason is that, although MA initially chooses the local optimum more often than SSWM, it still escapes quickly. As a result we designed a 5 state model which “locks” the algorithms to their initial choices. By amplifying the function to contain several copies of the 5 state model we achieve our goal of defining a composite function where SSWM is efficient while MA requires exponential time with overwhelming probability, independent from its temperature parameter.

Given the similarities between SSWM and other particularly selective strategies such as steepest ascent and single-trajectory algorithms using offspring populations, we compared the performance of SSWM and MA with BILS, FILS and a (1,λ) RLS. We rigorously showed that BILS excels on the composite function and experiments have shown that the (1,λ) RLS performs comparable to SSWM for big enough λ.

Part III

An Application of Stochastic Differential Equations to Evolutionary Algorithms

Chapter 8

Modelling Evolutionary Algorithms with Stochastic Differential Equations

This chapter is based on the following publications:

1. Paixão, T. and Pérez Heredia, J. (2017). An application of stochastic differential equations to evolutionary algorithms. *Proceedings of the 14th ACM/SIGEVO Conference on Foundations of Genetic Algorithms, FOGA '17*, pages 3–11. ACM.

2. Pérez Heredia, J. (2017). Modelling Evolutionary Algorithms with Stochastic Differential Equations. In *Evolutionary Computation*. To appear.

Part II of this thesis focused on applying the runtime analysis techniques introduced in Chapter 2 to the strong selection weak mutation evolutionary regime derived in Chapter 3. We saw how this approach yields many interesting and meaningful results for both the evolutionary computing and population genetics communities. We now want to consider if EC can borrow some of the tools used for the analysis of allele frequencies. Specifically, we will consider the use of the diffusion theory explained in Section 3.2 and Appendix B.

Nowadays the theoretical study of randomised heuristics mainly focuses on rigorous runtime analysis, however we can trace back literature about general models of evolutionary computation to Holland (1975). Roughly a decade later, Goldberg (1987) focused on modelling the evolving population of a genetic algorithm (GA). This was extended by Vose (1995) who used a Markov Chain approach and fixed points analysis. In the same spirit of dynamical systems, we can find an analysis for steady state GAs by Wright and Rowe (2001).

However, Schema theory was proven wrong (Altenberg, 1994) and Markov Chain approaches are, in general, intractable (He and Yao, 2003). The remaining mentioned techniques share a common rationale, the stochastic process describing an EA varies according to its expected change. The same idea is used in the so-called ordinary differential equation (ODE) method by Wormald (1995). Here, the EA is considered as a noisy version of an ODE that tracks the expected rate of change. This method has been applied several times in EC (see e.g. Yin et al., 1995 or Akimoto et al., 2012).

Due to this deterministic approach of the expected change, these methods disregard the random component of EAs. In this chapter we use a method based on stochastic differential equations. This approach naturally includes the stochasticity implicit in randomised heuristics through the well established tools from Itô calculus. The use of SDE has already been in EC for convergence analysis by Schaul (2012), however the results did not improve the state of the art.

Other modelling attempts such as the application of statistical physics by Prügel-Bennett and Shapiro (1994) yielded highly accurate results when modelling finite populations and fluctuations were also modelled by Prügel-Bennett (1997). However, this method requires that the algorithm's selection operator weights each search point according to the Boltzmann distribution. This is the case for algorithms like MA or SSWM but it will not hold in general. On the other hand, this method and some of the previously mentioned approaches were tackling more general and ambitious problems than what this chapter covers.

Stochastic differential equations have proven to be a powerful tool when dealing with stochastic processes, many examples can be found in the textbook by Øksendal (2003). SDEs have accomplished great achievements in physics, starting with the explanation of Brownian motion by Einstein (1905). In finance, specially for Black and Scholes (1973) work on options prices that yielded a Nobel Prize in Economics in 1997. As we saw in Section 3.2, the more related field of Population Genetics made used of the diffusion theory, which as we will see, is connected with SDEs. SDEs have been used in PG to track the frequency of genes in a population. This approach was used by Kimura (1957) and has ever since been an integral part of the so-called modern synthesis. Many classical results were obtained through this approach, including the steady state distribution of gene frequencies of a finite population under mutation and selection by Kimura (1964), and the probability of fixation of a gene by Kimura (1968) (see Chapter 3). Our work also contributes to the renewed interest of unifying EC and PG as in Paixão et al. (2015).

Building on top of the so-called *diffusion approximation* (see Section 3.2 and Appendix B), we derive a SDE that will be used to model the dynamics of EAs. For some scenarios we will be able to solve this equation analytically. We show that this is the case by obtaining equivalent statements to the well-known additive (Theorem 2.7) and multiplicative drift (Theorem 2.9). Furthermore we will derive a new version of the multiplicative drift, extending its applicability to non-elitist algorithms even when the drift is pushing the algorithm away from the optimum. The work we present here has several aims:

- improving the understating of EA dynamics,
- deriving analytic expressions for the expectation and the variance,
- introducing SDE to the field of EC and
- translation of drift theorems from runtime analysis to fixed budget analysis.

This chapter focuses on a fixed budget analysis, we refer to Section 2.4 and the papers mentioned there for a introduction and motivation to fixed budget.

We present what to our best knowledge is the first fixed budget analysis of non-elitist EAs. Furthermore, our formulation considers any selection strength. This constitutes an important advantage since drift theorems require to bound the process by a positive and monotonic non-increasing distance function, (see e.g. Doerr et al., 2012; He and Yao, 2001; Johannsen, 2010). In the case of non-elitist EAs, this implies increasing the selection strength such that worsening moves are not very likely to be accepted. However in our results the drift can even be negative and yet we will be able to provide the expected behaviour, and the variance.

The disadvantage of the SDE method, unlike runtime analysis or other fixed budget techniques, is that it requires performing two approximations. The first approximation consists of considering the system as a time-continuous process. In the second approximation, we will assume that the change of the process $X_{t+1} - X_t$ follows a distribution without higher statistical moments than the variance. Hence, this method, although is mathematically based, is not completely rigorous. Due to this reason, the last section of the chapter presents an extensive validation of these approximations. We will compare the obtained results against rigorous literature for runtime and fixed budget. In addition, we present a graphical comparison against the simulated results, and we will experimentally measure the error of the approximation.

8.1 Stochastic Differential Equations

This subsection contains the minimal knowledge of SDEs to be able to understand this chapter. For more extended and rigorous details we refer the reader to the textbook by Øksendal (2003). Stochastic differential equations are equations that deal with the behaviour of stochastic processes. We can define these processes as a collection of random variables in $\mathbb{R}$ over time t (i.e. $\{X_t\}_{t\geq 0}$), together with an underlying probability space $(\Omega, \mathscr{F}, P)$ that from now on we will take for granted.

The simplest stochastic process is probably the well-known white noise W_t. Its signal integrated over time will produce another famous process $B_t = \int W_t dt$ known as Brownian motion or Wiener process. Some of the characteristics of these processes are that they have 0 mean and their increments are independent and stationary. Furthermore, all the moments higher than the second are 0 (see e.g. Ross, 1996). We will focus on SDEs of the form

$$\begin{aligned} dX_t &= b(t,X_t)dt + \sigma(t,X_t)W_t dt \\ &= b(t,X_t)dt + \sigma(t,X_t)dB_t. \end{aligned} \tag{8.1}$$

This equation represents a process where the state change dX_t depends on both the time and the current position (which can also depend on the time) through the terms $b(t,X_t)$ and $\sigma(t,X_t)$. These are usually referred to as the drift and diffusion terms or coefficients. We can say that b tracks the expected evolution of the process, note that if $\sigma = 0$ (absence of noise) the drift coefficient simply becomes dX_t/dt. And σ collects the random behaviour by amplifying a white noise or Brownian motion term. After applying the usual integration rules one obtains a simple expression for the state after t iterations.

$$X_t = X_0 + \int_0^t b(s,X_s)ds + \int_0^t \sigma(s,X_s)dB_s. \tag{8.2}$$

The challenge now is to prove the existence of an integral over an stochastic process $\int f dB_s$ and develop its integration rules. This was done by Itô (1944, 1946, 1950, 1951): he proved the existence of this integral, worked on stochastic integral equations of the type (8.2) and applied his developments to the already existing field of SDE. The key idea of Itô's work is a stochastic extension of the Riemann-Stieltjes (1894) integral, which is itself, an extension of the standard Riemann integral for when a function is integrated with respect to another function. The other traditional formulation was given by Stratonovich (1966). In this work we will focus on stochastic processes that fit into the following definition of an Itô process (Definition 4.1.1 in Øksendal, 2003).

Definition 8.1 (1-dimensional Itô process). *A 1-dimensional Itô process is a stochastic process X_t on $(\Omega, \mathscr{F}, P)$ of the form*

$$\frac{dX_t}{dt} = b(t,X_t) + \sigma(t,X_t)W_t \tag{8.3}$$

where W_t denotes a white noise process, b is absolute integrable in t and σ is square integrable in t.

In the following we will refer as Itô integrals to integrals over a Brownian process $\int f dB_s$ (a formal definition can be found in e.g. Øksendal, 2003). These integrals will have the following properties (adapted from Theorem 3.2.1 in Øksendal, 2003).

Theorem 8.1 (Properties of the Itô integral)**.** *Let $f,g \in \mathscr{V}(0,T)$ and let $0 \leq S < U < T$. Then*

(i) $\int_S^T f dB_t = \int_S^U f dB_t + \int_U^T f dB_t$

(ii) $\int_S^T (cf+g) dB_t = c \cdot \int_S^T f dB_t + \int_S^T g dB_t$ *(c constant)*

(iii) $E\left[\int_S^T f dB_t\right] = 0$

Performing a variable change on a stochastic integral is a bit more involved than for ordinary integrals. The usual rules do not apply and we will have to use the following statement (Theorem 4.1.2 in Øksendal, 2003) which is known as Itô's formula.

Theorem 8.2 (Itô formula)**.** *Let X_t be a Itô process given by*

$$dX_t = b(t,X_t)dt + \sigma(t,X_t)dB_t.$$

Let $g(t,x)$ be twice continuously differentiable on $x \in \mathbb{R}$ and once in $t \in \mathbb{R}^+$. Then $Y_t = g(t,X_t)$ is again an Itô process, and

$$dY_t = \frac{\partial g}{\partial t}dt + \frac{\partial g}{\partial x}dX_t + \frac{1}{2}\frac{\partial^2 g}{\partial x^2} \cdot (dX_t)^2, \tag{8.4}$$

where $(dX_t)^2 = (dX_t) \cdot (dX_t)$ is computed according to the rules

$$dt \cdot dt = dt \cdot dB_t = dB_t \cdot dt = 0, \quad dB_t \cdot dB_t = dt. \tag{8.5}$$

Then, an alternative form is given by

$$dY_t = \left(\frac{\partial g}{\partial t} + b\frac{\partial g}{\partial x} + \frac{\sigma^2}{2}\frac{\partial^2 g}{\partial x^2}\right) \cdot dt + \sigma\frac{\partial g}{\partial x}dB_t. \tag{8.6}$$

Finally, another useful property specially when computing higher moments, such as the variance, is the Itô Isometry (Corollary 3.1.7 in Øksendal, 2003).

Theorem 8.3 (Itô Isometry)**.** *For all $f \in \mathscr{V}(S,T)$*

$$E\left[\left(\int_S^T f(X_t,t)dB_t\right)^2\right] = E\left[\int_S^T f(X_t,t)^2 dt\right].$$

8.2 The Diffusion Approximation

This section is very similar to Section (3.2) where we used diffusion theory to track the time evolution of the probability density $\phi(p,t)$ of the allele frequencies p. However, the notation varies due to a change in the variable of interest. We now wish to track the time evolution of the probability density $p(x,t)$ of a stochastic process X_t. To study this variable we start again from the Chapman-Kolmogorov equation (3.17), which for simplicity we rewrite here with the updated notation.

$$p(x,t+\Delta t) = \int \Delta(\delta \mid x) \cdot p(x-\delta,t) \cdot d\delta, \tag{8.7}$$

where $\Delta(\delta \mid x)$ is the transition probability of reaching x from $x-\delta$.

Now we perform the diffusion approximation which refers to the joint application of a:

- **Time-continuous approximation:** first order Taylor expansion of $p(x,t+\Delta t)$ around $\Delta t = 0$. By Taylor's theorem it carries an error of order $O(\Delta t)$.

- **Gaussian approximation:** second order Taylor expansion of $\Delta(\delta \mid x) \cdot (x-\delta,t)$ around $x-\delta = x$. By Taylor's theorem it carries an error of order $O(\delta^2)$.

While the time-continuous approximation is intrinsic to using differential (and not difference) equations for a time-discrete process, the Gaussian approximation will be more accurate when δ is small i.e. when the process usually moves between close by states.

It can be shown that, by introducing both approximations in (8.7) we arrive to the well-known diffusion equation. Again, for the sake of simplicity, the calculations are included in the Appendix B. This equation is known as the Fokker-Planck equation or the Forward Kolmogorov equation (see e.g. Karlin and Taylor, 1981).

$$\Delta t \cdot \frac{\partial p(x,t)}{\partial t} \approx -\frac{\partial}{\partial x} p(x,t) \cdot \mathrm{E}\left[\Delta\right] + \frac{1}{2} \cdot \frac{\partial}{\partial x^2} p(x,t) \cdot \mathrm{E}\left[\Delta^2\right]. \tag{8.8}$$

This is an equation for the time evolution of the probability density $p(x,t)$ of the random variable X_t. In principle, one could solve this equation to obtain a probability distribution at any particular time point. However, this is often impossible in practice. Instead, we deal directly with the SDE associated with the Fokker-Planck equation and take that as a model of our EA. The SDE corresponding to equation 3.18 will be an Itô process of the form (see e.g. Øksendal, 2003):

$$dX_t \approx \mathrm{E}\left[\Delta\right] dt + \sqrt{\mathrm{E}\left[\Delta^2\right]} dB_t$$

This is the central equation for this thesis. We will use it to describe the dynamics of evolutionary algorithms by computing its drift coefficient $\mathrm{E}[\Delta]$ and second moment or diffusion coefficient $\sqrt{\mathrm{E}[\Delta^2]}$. Finally, we present our model as an hypothesis where it is interesting to note that the first moment $E(\Delta)$ exactly corresponds to the drift as used in runtime analysis (see Subsection 2.3.4).

Hypothesis 8.1 (The Diffusion Approximation)**.** *The dynamics of any algorithm described by Algorithm 2.2 with local or global mutations can be approximated by an Itô process X_t of the form*

$$\begin{aligned} dX_t &= b(X_t,t)dt + \sigma(X_t,t)dB_t \\ b(X_t,t) &= E[X_{t+1} - X_t | X_t] \\ \sigma^2(X_t,t) &= E\left[(X_{t+1} - X_t)^2 | X_t\right], \end{aligned}$$

where B_t is a Brownian motion process.

8.2.1 Are Evolutionary Algorithms Diffusive Processes?

The previous hypothesis would be a rigorous theorem if evolutionary algorithms were a pure diffusion process, but they are not. First of all, we used differential and not difference equations to model a discrete process. This approximation is intrinsic to the chosen method and carries an error of constant size.

On the other hand, the Gaussian approximation depends on each algorithm and the fitness function. The variable of interest is the transition probability $\Delta(\delta \mid x)$ from Equation (8.7). Which reads as the probability of reaching the state x from the state $x - \delta$ in one iteration. It was shown that the Gaussian approximation will carry an error of size $O(\delta^2)$ (See Appendix B). Hence, depending on the evolutionary operators of the EA this approximation will or will not be accurate.

To motivate the use of this approximation for trajectory-based algorithms (see Section 2.2), we can focus just on the mutation operator. The reason is that selection cannot increase the distance δ between the parent and the child solutions. It will only decide if the new individual is accepted or rejected.

The question resides then in how big is the δ produced by mutation. To illustrate this, let us consider a random walk on the Boolean hypercube and its number of ones x. First, with local mutations, the RW will have a constant $\delta = 1$. On the other hand, if global mutations are used δ can be any integer number in $\{0, 1, \ldots, n\}$. Fortunately, the probability of large mutations decays exponentially with their size δ. Actually, the two smallest values of δ (0 and 1) will carry together a probability mass of approximately $2/e$. Hence, all the larger jumps together will only have a probability of $\approx 1-2/e$. Since δ is concentrated, the higher order moments should be close to zero, suggesting that the Gaussian approximation is also accurate for global mutations. However, on some fitness problems all the relevant events could belong to the neglected probability mass of $1-2/e$. In this chapter, we only consider problems (ONEMAX and LEADINGONES) where, as we will see, global mutations are decently Gaussian approximated. Nevertheless, at the end of the chapter, we provide an extensive validation for Hypothesis 8.1.

8.3 Drift Theorems for Fixed Budget Analysis

The Diffusion Approximation (Hypothesis 8.1) considers a scenario which in general does not have an analytic solution. This will be dependent on the mathematical expressions of the drift $b(X_t,t)$ and diffusion $\sigma(X_t,t)$ coefficients. In this chapter we will consider two simple scenarios where we can obtain equivalent results to drift theorems for runtime analysis.

The methods to solve SDEs are similar to those to solve ODE, but we have to use the Itô formula (Theorem 8.2) when performing a change of variables. In the following two subsections our approach is to use *a priori* knowledge. We will find another process Y_t, described by a simpler function $g(x,t)$ that we know is the solution of the equivalent ODE (SDE with $\sigma = 0$). For example, if our process is of the form $dX_t = X_t dt + X_t dB_t$ our candidate solution will be $Y_t = \ln(X_t)$. When the coefficients are more complex, we will try to find a variable change that leads to an already known SDE.

Before starting the derivation of these theorems, we would like to emphasise that this section is purely rigorous. No approximations are used unlike in the derivation of Hypothesis 8.1.

8.3.1 Additive Drift

The simplest case is when the drift $b(X_t,t)$ does not depend on the time t or the current state X_t i.e. $b(X_t,t) = b$, as in the additive drift for runtime analysis (Theorem 2.7).

Theorem 8.4. *Let X_t be an Itô process of the form $dX_t = bdt + \sigma dB_t$, where B_t is a Brownian process, $b \in \mathbb{R}$ and $\sigma \in \mathbb{R}_0^+$. Then,*

$$\begin{aligned} X_t &= X_0 + bt + \sigma B_t \\ E[X_t] &= E[X_0] + bt \\ Var[X_t] &= Var[X_0] + \sigma^2 t. \end{aligned}$$

Proof. This is a trivial case that can be solved by directly integrating

$$\int_0^t dX_s = \int_0^t b ds + \int_0^t \sigma dB_s \tag{8.9}$$

$$X_t = X_0 + bt + \sigma B_t \quad (B_0 = 0). \tag{8.10}$$

Taking expectations in Equation (8.9) we can compute the expected value

$$\mathrm{E}[X_t] = \mathrm{E}[X_0] + \mathrm{E}\left[\int_0^t b ds\right] + \mathrm{E}\left[\int_0^t \sigma dB_s\right]$$

due to property (iii) from Theorem 8.1 the last term averages out yielding

$$\mathrm{E}[X_t] = \mathrm{E}[X_0] + \int_0^t \mathrm{E}[b] \, ds \; = \mathrm{E}[X_0] + bt. \tag{8.11}$$

Finally, to compute the variance we use the results obtained in (8.10) and (8.11) as follows

$$\mathrm{Var}[X_t] = \mathrm{E}\left[(X_t - \mathrm{E}[X_t])^2\right] = \mathrm{E}\left[(X_0 - \mathrm{E}[X_0])^2 + \sigma^2 B_t^2\right] = \mathrm{Var}[X_0] + \sigma^2 \cdot \mathrm{E}\left[B_t^2\right].$$

Considering a normalised Brownian motion ($\mathrm{E}\left[B_t^2\right] = t$) yields the claimed statement. □

As in runtime analysis, it is not typically the case that we know exactly the drift's value, but we can bound it within some range. It follows directly that knowing those bounds we will be able to bound the expected progress after t iterations.

Corollary 8.1. *In the context of Theorem 8.4, if $b_\ell \leq b \leq b_u$ and $\sigma_\ell \leq \sigma \leq \sigma_u$, with $b_\ell, b_u \in \mathbb{R}$ and $\sigma_\ell, \sigma_u \in \mathbb{R}_0^+$. Then*

$$X_0 + b_\ell t + \sigma_l B_t \leq X_t \leq \; X_0 + b_u t + \sigma_u B_t$$
$$E[X_0] + b_\ell t \leq E[X_t] \leq \; E[X_0] + b_u t$$
$$Var[X_0] + \sigma_\ell^2 t \leq Var[X_t] \leq \; Var[X_0] + \sigma_u^2 t.$$

8.3.2 Multiplicative Drift

Analogous to runtime analysis, the assumption of additive drift is in general too loose since it does not take into account the current state of the process. This is solved by the multiplicative drift theorem (Theorem 2.9) when the drift is proportional to the current state X_t.

As discussed in the introduction, SDEs are a well established method in other research fields, allowing us to recycle already existing results. We will make use of a simplified version of the Cox-Ingersoll-Ross (CIR) model (see Cox et al., 1985; Maghsoodi, 1996) which describes the evolution of interest rates with the following SDE

$$dX_t = b(\theta - X_t)dt + \sigma\sqrt{X_t}dB_t. \tag{8.12}$$

The coefficient b specifies the speed (drift) at which the process approaches its expected value θ provoking a mean reversion effect. The noise due to the market dB_t is amplified by a term $\sigma\sqrt{X_t}$ to ensure that there are not negative interest rates (note that this effect is also obtained with any diffusion coefficient that goes to 0 when $X_t = 0$). We will use a simplified version without the mean reversion effect ($\theta = 0$), which resembles an elitist multiplicative drift process.

Theorem 8.5 (CIR without mean reversion Cox et al., 1985)**.** *Let $X_t > 0$ be an Itô process of the form*

$$dX_t = -bX_t dt + \sigma\sqrt{X_t}dB_t$$

where B_t is a Brownian process and $b \in \mathbb{R}^+$ and $\sigma \in \mathbb{R}_0^+$. Then,

$$\begin{aligned} E\left[X_t\right] &= E\left[X_0\right] \cdot e^{-bt} \\ Var\left[X_t\right] &= E\left[X_0\right] \frac{\sigma^2}{b} \cdot e^{-bt}\left(1 - e^{-bt}\right) \end{aligned}$$

Proof. The results for the expectation and variance of the original CIR process (8.12) are (Equation (19) in Cox et al., 1985)

$$\begin{aligned} \mathrm{E}\left[X_t\right] &= \mathrm{E}\left[X_0\right] \cdot e^{-bt} + \theta\left(1 - e^{-t}\right) \\ \mathrm{Var}\left[X_t\right] &= \mathrm{E}\left[X_0\right] \frac{\sigma^2}{b} \cdot e^{-bt}\left(1 - e^{-bt}\right) + \theta\frac{\sigma^2}{2b}\left(1 - e^{-t}\right)^2. \end{aligned}$$

Our statement is just the special case when $\theta = 0$. □

Once more, if we can only compute bounds rather than exact values it follows straightforward the next corollary. Notice that the result for the expectation is similar to a previous fixed-budget multiplicative drift (Theorem 2.16).

Corollary 8.2. *In the context of Theorem 8.5, if $b_\ell \leq b \leq b_u$ and $\sigma_\ell \leq \sigma \leq \sigma_u$ with $b_\ell, b_u \in \mathbb{R}^+$ and $\sigma_\ell, \sigma_u \in \mathbb{R}_0^+$. Then,*

$$\begin{aligned} E[X_0] \cdot e^{-b_\ell t} \geq\ E[X_t] &\geq E[X_0] \cdot e^{-b_u t} \\ Var[X_t] &\geq E[X_0] \frac{\sigma_\ell^2}{b_\ell} e^{-b_\ell t} \left(1 - e^{-b_\ell t}\right) \\ Var[X_t] &\leq E[X_0] \frac{\sigma_u^2}{b_u} e^{-b_u t} \left(1 - e^{-b_u t}\right). \end{aligned}$$

8.3.3 Non-elitist Multiplicative Drift

Drift theorems for the runtime share a common condition - the drift must be bounded by a monotonic function towards the optimum (see Theorems 2.7, 2.9, 2.11 and 2.16). In the following statement we relax this condition - the process also follows a monotonic behaviour in expectation, but towards the stationary distribution (which can be far away from the optimum). In fact, as we will see in the experimental section, even when the process starts near the optimum it will follow an exponential decay towards its equilibrium state, moving away from the optimum.

Although we have named it *Non-Elitist Multiplicative Drift* this theorem can also be applied for elitist algorithms. This will be the limit case when $b = \alpha = 0$ and $\beta < 0$ where we recover Theorem 8.5.

Theorem 8.6. *Let $X_t \geq 0$ be an Itô process of the form*

$$dX_t = a(b - X_t)dt + \sqrt{\alpha - \beta X_t} dB_t$$

with $a \in \mathbb{R}^+$, $b, \alpha \in \mathbb{R}_0^+$, $\beta \in \mathbb{R}$, $\alpha - \beta X_t \geq 0$ for all X_t, $\alpha - b\beta \geq 0$ and B_t is a Brownian process. Then,

$$\begin{aligned} X_t &= X_0 e^{-at} + b\left(1 - e^{-at}\right) + \int_0^t e^{-a(t-s)} \sqrt{\alpha - \beta X_s} dB_s \\ E[X_t \mid X_0] &= X_0 e^{-at} + b\left(1 - e^{-at}\right) \\ Var[X_t \mid X_0] &= \frac{\alpha}{2a}(1 - e^{-2at}) - \frac{\beta}{a} X_0 e^{-at}(1 - e^{-at}) - \frac{b\beta}{2a}\left(1 - e^{-at}\right)^2 \end{aligned}$$

Proof. Let us use the integrating factor e^{at} to perform the variable change $Y_t = e^{at}X_t$, then by applying Itô formula (Theorem 8.2) we obtain

$$\begin{aligned}
dY_t &= \left(\frac{\partial(e^{at}X_t)}{\partial t} + a(b-X_t)\frac{\partial(e^{at}X_t)}{\partial X_t} + \frac{\alpha-\beta X_t}{2}\frac{\partial^2(e^{at}X_t)}{\partial X_t^2}\right)\cdot dt \\
&\quad + \sqrt{\alpha-\beta X_t}\frac{\partial(e^{at}X_t)}{\partial X_t}dB_t \\
&= \left(ae^{at}X_t + a(b-X_t)e^{at}\right)\cdot dt + \sqrt{\alpha-\beta X_t}e^{at}dB_t \\
&= abe^{at}dt + \sqrt{\alpha-\beta X_t}e^{at}dB_t.
\end{aligned}$$

Applying the usual integration rules leads to

$$Y_t = Y_0 + ab\int_0^t e^{as}ds + \int_0^t \sqrt{\alpha-\beta X_s}e^{as}dB_s$$

Let us now compute just the first integral and also revert the variable change. Then,

$$e^{at}X_t = X_0 + b\left(e^{at}-1\right) + \int_0^t \sqrt{\alpha-\beta X_s}e^{as}dB_s.$$

Multiplying both sides by e^{-at} leads to the first claimed result in the theorem. For the second result we take expectations leading to

$$\mathrm{E}[X_t] = \mathrm{E}[X_0]e^{-at} + b\left(1-e^{-at}\right) + \mathrm{E}\left[\int_0^t e^{-a(t-s)}\sqrt{\alpha-\beta X_s}dB_s\right].$$

Fortunately, expectations of integrals over a Brownian motion average out (see property (iii) from Theorem 8.1) yielding the second result of the theorem. Finally, to compute the variance we will start from the formula $\mathrm{Var}[X_t] = \mathrm{E}\left[(X_t - \mathrm{E}[X_t])^2\right]$ which after introducing the two previous results yields

$$\mathrm{Var}[X_t \mid X_0] = \mathrm{E}\left[\left(\int_0^t \sqrt{\alpha-\beta X_s}e^{-a(t-s)}dB_s\right)^2\right]$$

applying Itô isometry (Theorem 8.3) we obtain

$$\mathrm{Var}[X_t \mid X_0] = \int_0^t (\alpha-\beta\mathrm{E}[X_s])e^{-2a(t-s)}ds = \alpha e^{-2at}\int_0^t e^{2as}ds - \beta e^{-2at}\int_0^t \mathrm{E}[X_s]e^{2as}ds$$

introducing the result for the expectation leads to

$$\begin{aligned}\mathrm{Var}[X_t \mid X_0] &= \alpha e^{-2at}\int_0^t e^{2as}ds - \beta e^{-2at}\int_0^t \left(X_0 e^{-as} + b\left(1-e^{-as}\right)\right)e^{2as}ds\\ &= \alpha e^{-2at}\int_0^t e^{2as}ds - \beta e^{-2at}X_0\int_0^t e^{as}ds - b\beta e^{-2at}\int_0^t \left(e^{2as}-e^{as}\right)ds\end{aligned}$$

computing the integrals completes the proof

$$\begin{aligned}\mathrm{Var}[X_t \mid X_0] &= \alpha e^{-2at}\cdot\frac{e^{2at}-1}{2a} - \beta e^{-2at}X_0\cdot\frac{e^{at}-1}{a} - b\beta e^{-2at}\cdot\frac{\left(e^{at}-1\right)^2}{2a}\\ &= \frac{\alpha}{2a}(1-e^{-2at}) - \frac{\beta}{a}X_0 e^{-at}(1-e^{-at}) - \frac{b\beta}{2a}\left(1-e^{-at}\right)^2. \qquad \square\end{aligned}$$

Once more, if exact values for the drift and diffusion coefficients are hard to compute, but can be bounded we can make use of the following corollary.

Corollary 8.3. *In the context of Theorem 8.6, if $a_\ell(b_\ell - X_t) \le a(b-X_t) \le a_u(b_u - X_t) \in R^+$ for all X_t, and $\alpha_\ell - \beta_\ell X_t \le \alpha - \beta X_t \le \alpha_u - \beta_u X_t \in R_0^+$ for all X_t. Then,*

$$\begin{aligned}X_0 e^{-a_\ell t} + b_\ell\left(1-e^{-a_\ell t}\right) \le E[X_t \mid X_0] &\le X_0 e^{-a_u t} + b_u\left(1-e^{-a_u t}\right)\\ Var[X_t \mid X_0] &\le \frac{\alpha_u}{2a_u}(1-e^{-2a_u t}) - \frac{\beta_u}{a_u}X_0 e^{-a_u t}(1-e^{-a_u t}) - \frac{b_u\beta_u}{2a_u}\left(1-e^{-a_u t}\right)^2\\ Var[X_t \mid X_0] &\ge \frac{\alpha_\ell}{2a_\ell}(1-e^{-2a_\ell t}) - \frac{\beta_\ell}{a_\ell}X_0 e^{-a_\ell t}(1-e^{-a_\ell t}) - \frac{b_\ell\beta_\ell}{2a_\ell}\left(1-e^{-a_\ell t}\right)^2.\end{aligned}$$

8.4 Applications

Although the results derived in the previous section are rigorous, its application for evolutionary algorithms is just an approximation (see Section 8.2). In this section we will take for granted that the diffusion approximation holds and we will relax our statements to the category of *application* rather than *theorem*. It will be finally in section 8.5 where we will extensively validate this assumption (Hypothesis 8.1).

8.4.1 Elitist Algorithms on LeadingOnes

First, we start with a study case for the additive drift (Theorem 8.4). A good problem candidate, although not ideal, is the LeadingOnes function. Many algorithms like RLS or the (1+1) EA have an *almost* additive drift on this function, however at the end of the optimisation process the drift depends on the state. The reason this fitness function is not ideal for an additive drift approach are the fitness boundary conditions ($0 \leq x \leq n$). Our formulation of additive drift does not consider boundaries on the process. SDEs can cope with these conditions but this will highly increase the complexity and extension of the chapter deviating from our introductory interests. To circumvent this problem, we will add a clause in the application's statement excluding our results after the optimum is reached. This approach has already been used in the literature as in Nallaperuma et al. (2017a).

Application 8.1. *Under the Diffusion Approximation, after t iterations, RLS on* LEADINGONES *will find the optimum or reach an expected value and variance of:*

$$\begin{gathered}
E[X_0] + \frac{1}{n}t \leq E[X_t] \leq E[X_0] + \frac{2}{n}t \\
E[X_0] + \frac{1}{n}t \leq Var[X_t] \leq Var[X_0] + \frac{4}{n}t
\end{gathered}$$

where X_0 is the number of leading ones of the initial search point.

Proof. Let us denote by X_t the number of leading ones in the bitstring at time t. And assume that Hypothesis 8.1 is a good approximation for the dynamics of X_t. As explained in the introduction, the drift and diffusion coefficients will be the first and second moments of the rate of change of the process. Or mathematically speaking

$$\begin{aligned}
b &= \mathrm{E}[X_{t+1} - X_t \mid X_t] = \sum_x x \cdot p(X_{t+1} - X_t = x \mid X_t) \\
\sigma^2 &= \mathrm{E}\left[(X_{t+1} - X_t)^2 \mid X_t\right] = \sum_x x^2 \cdot p((X_{t+1} - X_t)^2 = x \mid X_t).
\end{aligned}$$

Since RLS uses local mutations it can only produce points in the Hamming neighbourhood, with only one out of these n points (flipping the first 0-bit) leading to an improvement. Note that this event will in fact increase X_t in at least 1, actually the increment will be almost 2 due to the geometric distribution followed by the free riders (bits after the leading ones), see e.g. Lehre and Witt (2012b). Hence, the drift coefficient will be the probability of mutation sampling an improving search point times the fitness increment of that mutation.

$$b(X_t) = \frac{1}{n} \cdot \left(2 - 2^{-(n-1-X_t)}\right).$$

We now bound the drift by bounding the impact in the range $[1, 2)$ yielding

$$b_l = \frac{1}{n}, \quad b_u = \frac{1}{n} \cdot 2. \tag{8.13}$$

To compute the diffusion coefficient σ we use the same rationale but raising the impact to the power of 2, therefore

$$\sigma_l^2 = \frac{1}{n} \quad \sigma_u^2 = \frac{1}{n} \cdot 2^2.$$

These coefficient bounds fit in our additive drift description (Corollary 8.1), using this corollary with the previously computed coefficients directly leads to the claimed results. □

The application to the (1+1) EA is equivalent but we have to deal with global mutations, which makes computing the exact value for the drift and diffusion coefficients harder.

Application 8.2. *Under the Diffusion Approximation, after t iterations, the (1+1) EA on* LEADINGONES *will find the optimum or reach an expected value and variance of:*

$$E[X_0] + \frac{1}{en}t \leq E[X_t] \leq E[X_0] + \frac{2}{n}t$$
$$Var[X_0] + \frac{1}{en}t \leq Var[X_t] \leq Var[X_0] + \frac{4}{n}t$$

where X_0 is the number of leading ones of the initial search point.

Proof. Let us denote by X_t the number of leading ones in the bitstring at time t. And assume that Hypothesis 8.1 is a good approximation for the dynamics of X_t. As in the previous application, the drift can be expressed with two terms: the probability of having a positive drift and its impact. The impact term remains identical, but the mutational term is different since we have to ensure that all the 1-bits of the current solution are preserved.

$$b(X_t) = \frac{1}{n}\left(1-\frac{1}{n}\right)^{X_t} \cdot \left(2-2^{-(n-1-X_t)}\right).$$

To obtain the bounds for the drift we will consider the extreme cases $X_t = 0$ and $X_t = n-1$ for the mutational term and bound the impact in the range $[1,2)$ yielding

$$b \leq \frac{1}{n} \cdot 2 = b_u \tag{8.14}$$
$$b \geq \frac{1}{n}\left(1-\frac{1}{n}\right)^{n-1} \cdot 1 \geq \frac{1}{en} = b_l. \tag{8.15}$$

For the diffusion coefficient we can recycle the previous probabilities but we have to raise the impact to the power of 2 to obtain the second moment of the process

$$\sigma^2 \geq \frac{1}{en} \cdot 1^2 = \sigma_l^2,$$
$$\sigma^2 \leq \frac{1}{n} \cdot 2^2 = \sigma_u^2.$$

Finally, introducing these calculations in Corollary 8.1 proves the result. □

8.4.2 Elitist Algorithms on OneMax

Secondly we apply the multiplicative drift for the same algorithms but for the OneMax, problem which is ideal for this drift theorem. Unlike in LeadingOnes, the boundary conditions are not a problem since our multiplicative drift bounds can not push the system out of the fitness interval $[0,n]$. Notice that the drift is always increasing the number of ones and vanishes when approaching the optimum.

Application 8.3. *Under the Diffusion Approximation, the expected fitness reached by RLS in* OneMax *after t iterations is*

$$E[X_t] = n\left(1 - e^{-\frac{t}{n}}\right) + E[X_0] \cdot e^{-\frac{t}{n}}$$
$$Var[X_t] = (n - E[X_0]) \cdot e^{-\frac{t}{n}}\left(1 - e^{-\frac{t}{n}}\right)$$

where $E[X_0]$ is the expected number of ones of the initial search point.

Proof. Let us denote by Z_t the number of ones in the bitstring at time t. And assume that Hypothesis 8.1 is a good approximation for the dynamics of Z_t. The drift coefficient, as always, will be the first moment (expectation) of the process. The diffusion coefficient will be the second moment.

$$\begin{aligned} b(Z_t) &= \mathrm{E}[Z_{t+1} - Z_t \mid Z_t] \\ &= \sum_z z \cdot p(Z_{t+1} - Z_t = z \mid Z_t), \\ \sigma^2(Z_t) &= \mathrm{E}\left[(Z_{t+1} - Z_t)^2 \mid Z_t\right] \\ &= \sum_z z^2 \cdot p((Z_{t+1} - Z_t)^2 = z \mid Z_t). \end{aligned}$$

In the ONEMAX problem at state Z_t there are Z_t bit-flips out of n that lead to an improvement, with each of these events reducing the number of zeros by 1, therefore

$$b(Z_t) = -\frac{Z_t}{n}.$$

For the diffusion coefficient we just have to repeat the calculations raising to 2 the impact of each event $(-1)^2$, yielding

$$\sigma^2(Z_t) = (-1)^2 \frac{Z_t}{n} = \frac{Z_t}{n}.$$

Therefore the approximated SDE for this process is of the form $dZ_t = b(Z_t)dt + \sigma\left(\sqrt{Z_t}\right)dB_t$, which fits in our multiplicative drift (Theorem 8.5). Applying this theorem with $b = -1/n$ and $\sigma = 1/\sqrt{n}$ we obtain the following results

$$\begin{aligned} \mathrm{E}[Z_t] &= \mathrm{E}[Z_0] \cdot e^{-\frac{t}{n}} \\ \mathrm{Var}[Z_t] &= \mathrm{E}[Z_0] \cdot e^{-\frac{t}{n}} \left(1 - e^{-\frac{t}{n}}\right). \end{aligned}$$

Finally translating to the number of ones $X_t = n - Z_t$ leads to the theorem's statement. □

The next application is regarding the (1+1) EA, again in this case we have computed bounds on the coefficients rather than exact values due to the difficulty to obtain exact results with global mutations.

Application 8.4. *Under the Diffusion Approximation, the expected fitness reached by the (1+1) EA in* ONEMAX *after t iterations is bounded by*

$$\begin{aligned} E[X_t] &\geq n\left(1 - e^{-\frac{t}{en}}\right) + E[X_0] \cdot e^{-\frac{t}{en}} \\ E[X_t] &\leq n\left(1 - e^{-\frac{3.1t}{n}}\right) + E[X_0] \cdot e^{-\frac{3.1t}{n}} \\ Var[X_t] &\geq (n - E[X_0]) \cdot e^{-\frac{t}{en}} \left(1 - e^{-\frac{t}{en}}\right) \\ Var[X_t] &\leq (n - E[X_0]) \cdot e^{-3.1\frac{t}{n}} \left(1 - e^{-3.1\frac{t}{n}}\right) \end{aligned}$$

where $E[X_0]$ is the expected number of ones of the initial search point.

Proof. Let us denote by Z_t the number of ones in the bitstring at time t. And assume that Hypothesis 8.1 is a good approximation for the dynamics of Z_t. The drift and diffusion coefficients (first and second statistic moments of the process) can be expressed as

$$\begin{aligned} b(Z_t) &= \sum_{j=1}^{Z_t} -j \cdot \mathrm{mut}(Z_t, Z_t - j) \\ \sigma^2(Z_t) &= \sum_{j=1}^{Z_t} (-j)^2 \cdot \mathrm{mut}(Z_t, Z_t - j), \end{aligned}$$

where $\mathrm{mut}(Z_t, Z_t - j)$ is the probability of a mutation moving the process from Z_t to $Z_t - j$ zeros.

For such event to occur, we need that k 1-bits are flipped into 0-bits, and that $k+j$ 0-bits are flipped into 1-bits. Then,

$$\mathrm{mut}(Z_t, Z_t - j) = \sum_{k=0}^{n} \binom{n - Z_t}{k} \binom{Z_t}{k+j} \left(\frac{1}{n}\right)^{j+2k} \left(1 - \frac{1}{n}\right)^{2n-j-2k}.$$

The drift coefficient can be lower bounded using only the case when $j = 1$ leading

$$b(Z_t) \geq -\frac{Z_t}{n}\left(1 - \frac{1}{n}\right)^{n-1} \geq -\frac{Z_t}{ne}.$$

For the upper bound, we make use of Lemma 2.1 to upper bound $\mathrm{mut}(Z_t, Z_t + j)$ by $\left(\frac{Z_t}{n}\right)^j \cdot \left(1 - \frac{1}{n}\right)^{n-j} \cdot \frac{1.14}{j!}$ yielding

$$\begin{aligned} b(Z_t) &= -\sum_{j=1}^{Z_t} \left(\frac{Z_t}{n}\right)^j \cdot \left(1 - \frac{1}{n}\right)^{n-j} \cdot \frac{1.14}{j!} \cdot j \\ &\leq -1.14 \cdot \frac{Z_t}{n} \cdot \sum_{j=1}^{\infty} \frac{j}{j!} \\ &= -1.14 \cdot \frac{Z_t}{n} \cdot e \\ &\leq -3.1 \cdot \frac{Z_t}{n}. \end{aligned}$$

Analogous calculations lead to the following bounds on the diffusion coefficient

$$\frac{Z_t}{en} \leq \sigma^2(Z_t) \leq 3.1 \cdot \frac{Z_t}{n}.$$

Calling Corollary 8.2 with $-1/en \leq b \leq -3.1/n$ and $1/en \leq \sigma^2 \leq 3.1/n$ leads to

$$\begin{aligned}
\mathrm{E}[Z_t] &\geq \mathrm{E}[Z_0] \cdot e^{-\frac{1}{en}t} \\
\mathrm{E}[Z_t] &\leq \mathrm{E}[Z_0] \cdot e^{-\frac{3.1}{n}t} \\
\mathrm{Var}[Z_t] &\geq \mathrm{E}[Z_0] \cdot e^{-\frac{t}{en}} \left(1 - e^{-\frac{t}{en}}\right) \\
\mathrm{Var}[Z_t] &\leq \mathrm{E}[Z_0] \cdot e^{-3.1\frac{t}{n}} \left(1 - e^{-3.1\frac{t}{n}}\right).
\end{aligned}$$

Finally translating to the number of ones $X_t = n - Z_t$ leads to the claimed results. □

8.4.3 Non-Elitist Algorithms on OneMax

In this subsection we will consider algorithms with local mutation but different selection operators. We keep general the acceptance probability, with the only assumption that it is monotonically non-decreasing. Afterwards, in the validation section we will plug in the expressions for a RW, MA and SSWM.

Lemma 8.1. *Consider any Algorithm 2.3 with a monotonically non-decreasing acceptance function $p_{\text{acc}}(\Delta f)$ on* OneMax. *Then,*

$$\begin{aligned}
E[X_{t+1} - X_t \mid X_t] &= \frac{p^{\uparrow} + p^{\downarrow}}{n} \cdot \left(n \frac{p^{\uparrow}}{p^{\uparrow} + p^{\downarrow}} - X_t\right) \\
E\left[(X_{t+1} - X_t)^2 \mid X_t\right] &= p^{\uparrow} - \frac{X_t}{n}\left(p^{\uparrow} - p^{\downarrow}\right).
\end{aligned}$$

where X_t denotes the number of 1*-bits in the search point at time t, $p^{\uparrow} = p_{\text{acc}}(\Delta f = 1)$ and $p^{\downarrow} = p_{\text{acc}}(\Delta f = -1)$.*

Proof. Let us start from the definition of expectation

$$\mathrm{E}[X_{t+1} - X_t \mid X_t] = \sum_x x \cdot p(X_{t+1} - X_t = x \mid X_t)$$

now we use the same arguments as for the analysis of RLS on OneMax (see the proof of Application 8.3) but considering the acceptance of deleterious mutations. Then,

$$\mathrm{E}[X_{t+1} - X_t \mid X_t] = \frac{n - X_t}{n} p^{\uparrow} - \frac{X_t}{n} p^{\downarrow} = p^{\uparrow} - \frac{X_t}{n}\left(p^{\uparrow} + p^{\downarrow}\right)$$

multiplying and dividing by $\left(p^{\uparrow}+p^{\downarrow}\right)/n$ leads to

$$\mathrm{E}\left[X_{t+1}-X_t \mid X_t\right]=\frac{p^{\uparrow}+p^{\downarrow}}{n}\cdot\left(n\frac{p^{\uparrow}}{p^{\uparrow}+p^{\downarrow}}-X_t\right).$$

For the diffusion coefficient, we use the definition of the second statistical moment

$$\mathrm{E}\left[(X_{t+1}-X_t)^2 \mid X_t\right]=\sum_x x^2\cdot p(X_{t+1}-X_t=x \mid X_t)$$

using the same rationale as for the drift coefficient but noticing that the second moment can only change by $(\pm 1)^2$ yields

$$\mathrm{E}\left[(X_{t+1}-X_t)^2 \mid X_t\right]=(+1)^2\cdot\frac{n-X_t}{n}p^{\uparrow}+(-1)^2\cdot\frac{X_t}{n}p^{\downarrow}=p^{\uparrow}-\frac{X_t}{n}\left(p^{\uparrow}-p^{\downarrow}\right). \qquad \square$$

Once the coefficients are known, the following application derives the fixed budget results by using the non-elitist multiplicative drift (Theorem 8.6).

Application 8.5. *Consider any Algorithm 2.3 with a monotonically non-decreasing acceptance function* $p_{\mathrm{acc}}(\Delta f)$ *on* ONEMAX. *Under the Diffusion Approximation the following statements hold*

$$\begin{aligned}
E[X_t \mid X_0] &= X_0 e^{-\frac{p^{\uparrow}+p^{\downarrow}}{n}t}+n\frac{p^{\uparrow}}{p^{\uparrow}+p^{\downarrow}}\left(1-e^{-\frac{p^{\uparrow}+p^{\downarrow}}{n}t}\right)\\
Var[X_t \mid X_0] &= \frac{np^{\uparrow}}{2(p^{\uparrow}+p^{\downarrow})}\left(1-e^{-2t\frac{p^{\uparrow}+p^{\downarrow}}{n}}\right)-\frac{p^{\uparrow}-p^{\downarrow}}{p^{\uparrow}+p^{\downarrow}}X_0 e^{-\frac{p^{\uparrow}+p^{\downarrow}}{n}t}\left(1-e^{-\frac{p^{\uparrow}+p^{\downarrow}}{n}t}\right)\\
&\quad-\frac{p^{\uparrow}(p^{\uparrow}-p^{\downarrow})}{2n}\left(1-e^{-\frac{p^{\uparrow}+p^{\downarrow}}{n}{}_l t}\right)^2
\end{aligned}$$

where X_t *denotes the number of* 1*-bits in the search point at time t,* $p^{\uparrow}=p_{\mathrm{acc}}(\Delta f=1)$ *and* $p^{\downarrow}=p_{\mathrm{acc}}(\Delta f=-1)$.

Proof. Assume that Hypothesis 8.1 is a good approximation for the dynamics of X_t. Then, the statement is proven after a direct application of Theorem 8.6 with the coefficients from Lemma 8.1.

$$a=\frac{p^{\uparrow}+p^{\downarrow}}{n},\quad b=n\frac{p^{\uparrow}}{p^{\uparrow}+p^{\downarrow}},\quad \alpha=p^{\uparrow},\quad \beta=\frac{p^{\uparrow}-p^{\downarrow}}{n}. \tag{8.16}$$

$\square$

Algorithm	$\mathrm{E}[X_t]$	Literature	Diffusion Approximation
RLS	$\geq$	$1+2t/n-2^{-\Omega((1-\beta)n)}$	$1+t/n-2^{-n}$
	$\leq$	$1+2t/n-2^{-n}$	$1+2t/n-2^{-n}$
(1+1) EA	$\geq$	$1+2t/n-o(t/n)$	$1+t/(en)-2^{-n}$
	$\leq$	$1+2t/n-o(t/n)$	$1+2t/n-2^{-n}$

Table 8.1 Comparison of results for LEADINGONES. The *literature* column is obtained from Theorems 7 and 8 in Jansen and Zarges (2012). The results for RLS hold for $t=(1-\beta)n^2$ with $1/2+c<\beta<1$. In the case of the (1+1) EA, the result holds for $t=\frac{(1-\beta)n^2}{\alpha(n)}$ with $1/2+c<\beta<1$ and $\alpha(n)=w(1)$. The *diffusion approximation* column corresponds with Applications 8.1 and 8.2 and hold for $t<\min\{t\geq 0 \mid X_t=n\}$. Random initialisation was considered in all the cases: $\mathrm{E}[X_0]=1-2^{-n}$.

8.5 Validation of the Diffusion Approximation

The previous section showed several case studies of the application of the three rigorous drift theorems for fixed budget analysis. However this method was the result of an approximation which we were assuming to be accurate (Hypothesis 8.1). Finally, we provide an extensive validation to this assumption. We will tackle the accuracy of the method from several angles including experiments but also comparing the results against those obtained by rigorous methods.

8.5.1 Comparison with the Literature

We now compare the results obtained in subsections 8.4.1 and 8.4.2 with some of the fixed budget literature. For the comparison on the LEADINGONES problem, we use the paper that presented the first fixed budget analysis (Jansen and Zarges, 2012). In Table 8.1, we observe a high agreement for the upper bounds estimations (even an exact match for the RLS). Both approaches obtained the same growth term of $2t/n$ and the discrepancies for the (1+1) EA are present only on small terms. However, on the lower bounds the discrepancy is bigger, our method only obtained a growth term of t/n since we had to pessimistically use the smallest drift value (when $X_t=n-1$) to be able to apply the additive drift (recall the proof of Application 8.1). On the other hand, the results from the literature only hold for time budget values which are sufficiently smaller that the expected optimisation, whereas our results hold until the optimum is found. In Table 8.2 we consider the ONEMAX problem. Again, for RLS we compare against Jansen and Zarges (2012), we observe how both results (first two rows) can be recovered from our method noticing that $e^{-t/n}=(e^{-1/n})^t\approx(1-1/n)^t$, which is a

Algorithm	$\mathrm{E}[X_t]$	Literature	Diffusion Approximation
RLS	$=$	$n\left(1-(1-1/n)^t\right)$	$n\left(1-e^{-t/n}\right)$
	$=$	$n-n/2\,(1-1/n)^t$	$n-n/2e^{-t/n}$
(1+1) EA	$\geq$	$n\left(1-\frac{1}{2}e^{-t/(en)}\right)$	$n\left(1-\frac{1}{2}e^{-t/(en)}\right)$
	$\leq$	$n\left(1-\frac{1}{2}e^{-t/n}\right)-O(t/n)$	$n\left(1-\frac{1}{2}e^{-(3.1t)/n}\right)$

Table 8.2 Comparison of results for OneMax. The *literature* column is obtained from Theorems 4 and 5 in Jansen and Zarges (2012) and Theorem 5 in Lengler and Spooner (2015). The *diffusion approximation* column corresponds with Applications 8.3 and 8.4. Random initialisation was considered in all the cases apart from the first row ($X_0 = 0^n$).

good approximation for large n. In the case of the (1+1) EA we were able to obtain exactly the same lower bound as Lengler and Spooner (2015). The upper bounds are also in high agreement, however the constant inside the exponential term is not as tight as in the literature. Finally, since to our best knowledge, there are no previous fixed budget analysis of non-elitist algorithm we could not perform this sanity check for RW, MA and SSWM. Nonetheless, the following subsections circumvents this problem by recurring to the concept of stationary distribution.

8.5.2 Stationary Distribution

One feature of non-elitist EAs is that although they can sample the optimum, they might leave it and move to a search point of lower fitness. Actually, in the long run the algorithm (under certain conditions) will reach the so-called *stationary distribution* $\pi(x)$ for when the probabilities of leaving each state cancel out with the probabilities of reaching that state (see Subsection 2.3.1). Once the process reaches this limit state all the points in the search space will have a non-zero probability of being found when measuring the system. However a smart algorithm will assign higher probability masses to points of higher fitness.

If we consider now a RW with local mutations, it will follow the mutational bias towards bitstrings with $n/2$ zeros and $n/2$ ones. Then, after enough time depending on the initialisation, the expected number of ones will be $n/2$. In the case of MA and SSWM, we know from Theorems 2.2 and 7.1 that they share the following stationary distribution.

$$\pi(x) = \frac{e^{\gamma f(x)}}{Z} \tag{8.17}$$

where $\gamma = \alpha$ for MA, $\gamma = 2\beta(N-1)$ for SSWM and Z is the normalising constant.

Theorem 8.7. *Consider the Metropolis algorithm with $\gamma = \alpha$ and SSWM with $\gamma = 2\beta(N-1)$ on* OneMax *once they have reached the stationary distribution. Then for any initialisation, the expected number of ones is:*

$$E[X_t] = \frac{n}{1+e^{-\gamma}}.$$

Proof. In the case of OneMax we can express $\mathrm{E}[X_t]$ with the probabilities of each bit i being in each of the two possible values $p_i(1)$ and $p_i(0)$,

$$\mathrm{E}[X_t] = \sum_{i=1}^{n} \frac{p_i(1)}{p_i(1)+p_i(0)} = n \cdot \frac{p(1)}{p(1)+p(0)},$$

where in the last equality we have used the fact that each bit is optimised independently, thus we can apply linearity of expectations and consider that for any bit $p_i(1) = p(1)$ and $p_i(0) = p(0)$. Finally, introducing the probabilities at equilibrium from Equation (8.17) yields the claimed results. □

As discussed earlier, the long term (limit for $t \to \infty$) expectation of the non-elitist multiplicative drift tends to the b term from the drift coefficient which according to Equation (8.16) has a value of $b = (np^{\uparrow})/(p^{\uparrow}+p^{\downarrow})$. We can observe in the following table how this value matches exactly the real expected fitness at equilibrium $\mathrm{E}[X_t]$ showing the strength of the SDE approach for this scenario.

	RW	**MA**	**SSWM**
$p^{\uparrow}$	1	1	$\frac{1-e^{-2\beta}}{1-e^{-2N\beta}}$
$p^{\downarrow}$	1	$e^{-\alpha}$	$\frac{e^{2\beta}-1}{e^{2N\beta}-1}$
b	$n/2$	$\frac{n}{1+e^{-\alpha}}$	$\frac{n}{1+e^{-2\beta(N-1)}}$
$\mathrm{E}[X_t]$	$n/2$	$\frac{n}{1+e^{-\alpha}}$	$\frac{n}{1+e^{-2\beta(N-1)}}$

Table 8.3 Expected value at equilibrium $\mathrm{E}[X_t]$ from Theorem 8.7 and long term expectation b from Application 8.5 on OneMax. With $p^{\uparrow} = p_{\mathrm{acc}}(1)$ and $p^{\downarrow} = p_{\mathrm{acc}}(-1)$.

8.5.3 Reconciling Fixed Budget with Runtime Analysis

As argued by Jansen and Zarges (2012), the fixed budget perspective provides more insight on the optimisation process. Runtime analysis focuses on the expected first hitting time of the optimum, this time is expressed as a function of the problem size $\mathrm{E}[T] = f(n)$. On the other hand, fixed budget considers the expected fitness obtained after t iterations. Hence, the result will also depend on the time $\mathrm{E}[X_t] = f(n,t)$.

However, both approaches should be in concordance. If we know the first hitting time $T(X)$ of any specific fitness value X, it looks natural to compute the inverse function to obtain the equivalent fixed budget result. As shown by Doerr et al. (2013) this is the case for deterministic algorithms, but it will not be true in general for randomised algorithms. The authors circumvent this problem by using sharp bounds on the probability of deviating from the expected times. Finally, they developed a method to recycle expected optimisation times to derive fixed budget results.

In this section, we consider the inverse approach i.e. deriving optimisation times from fixed budget results. Again, inverting the function $\mathrm{E}[X_t] = f(n,t)$ for the time will not give in general the correct answer. However, we do not aim for a rigorous translation tool as achieved by Doerr et al. (2013). Our purpose is to obtain some time measurement that can be compared with the runtime literature (up to Landau's notation, Definition 2.8).

Additive Drift

Before jumping into conclusions for the studied algorithms, let us consider an additive process. This is, a collection of stochastic variables in time $X_{t\geq 0} \in \mathbb{R}$ whose expected change remains constant $\mathrm{E}[X_t - X_{t-1} \mid X_t] = b$. When b is positive, the expectation $\mathrm{E}[X_t]$ is constantly being increased and $\mathrm{E}[X_t]$ always reaches any finite target value x (given enough, but finite time). Let us recall that this will not be the case for EAs reaching the optimum x_{opt} of a fitness function, here $X_t \leq x_{\text{opt}}$ and therefore $\mathrm{Prob(X_t = x_{opt})} = 1$ is required in order for $\mathrm{E}[X_t] = x_{\text{opt}}$. We will consider first a pure additive process and afterwards we will relate it with RLS and the (1+1) EA optimising LEADINGONES.

We mentioned before that computing the inverse function of $\mathrm{E}[X_t]$ for the time t will not yield, in general, the correct result. However, this approach is correct for an additive drift process. Here, we can solve $\mathrm{E}[X_t] = f(n,t)$ for the time, obtaining a function of the form $t = f(\mathrm{E}[X], n)$. Finally, if we introduce any target value for $\mathrm{E}[X]$, it will yield the time when the expectation reaches the target value.

Theorem 8.8. *Consider a stochastic process $X_{t\geq 0} \in \mathbb{R}$ that follows an Additive Drift i.e. $E[X_t - X_{t-1} \mid X_t] = b$ for some constant $b > 0$. The time $\tau(x)$ is defined as the time when the expectation $E[X_t]$ reaches a value of x and is given by $\tau(x) = (x - E[X_0])/b$.*

Proof. By using the additive drift condition, we can compute $\mathrm{E}[X_t \mid X_0]$ as follows

$$\mathrm{E}[X_t \mid X_0] = X_0 + \sum_{i=0}^{t} \mathrm{E}[X_{i+1} - X_i \mid X_i] = X_0 + \sum_{i=0}^{t} b = X_0 + bt.$$

Using the law of total expectation, we obtain $\mathrm{E}[X_t] = \mathrm{E}[X_0] + bt$. Introducing $\mathrm{E}[X_t] = x$ and $t = \tau$ leads to $x = \mathrm{E}[X_0] + b\tau$, which solving for τ yields the claimed statement. □

It is straightforward to notice that this theorem is not applicable for optimisation algorithms. If X_t describes the fitness of the EA, we observe that the boundary condition $X_t \leq x_{\text{opt}}$ breaks the additive requirement: $\mathrm{E}\left[X_t - X_{t-1} \mid X_t = x_{\text{opt}}\right] \leq 0 \neq b$. As we mentioned earlier, we do not aim for a rigorous translation tool. However, as shown in Subsection 8.4.1 we can approximate the optimisation process of some algorithms on LeadingOnes with additive drift processes.

In Applications 8.1 and 8.2, we found additive drift bounds for the real drift of RLS and the (1+1) EA on LeadingOnes respectively. By introducing $x = n$ (the optimal fitness of LeadingOnes) in Theorem 8.8, we will compare, in a non-rigorous way, the $\tau(n)$-times for these processes with the runtimes from the literature. The following table presents these results showing a high agreement, up to smaller terms, of the SDE method with the literature. Since we used bounds for the drift coefficient b, the following table includes and upper τ_u and a lower τ_ℓ bound for the time $\tau(n)$.

LeadingOnes	**RLS**	**(1+1) EA**
b_ℓ	$1/n$	$1/en$
b_u	$2/n$	$2/n$
$\tau_\ell(n)$	$n^2/2$	$n^2/2$
$\tau_u(n)$	n^2	en^2
$\mathrm{E}[T]$	$\Theta(n^2)$	$\Theta(n^2)$

Table 8.4 Expected optimisation times $\mathrm{E}[T]$ from the literature (see e.g. Oliveto and Yao, 2011) and bounds on $\tau(n)$ from Theorem 8.8 for LeadingOnes. The bounds on the drift $b_{u,\ell}$ are taken from Equation (8.13) (RLS) and Equations (8.15) and (8.14) (lower and upper bound of the (1+1) EA respectively). Initialisation at 0^n.

Multiplicative Drift

In the case of the multiplicative drift (Theorems 8.5 and 8.6) if we were to perform the same sanity check we would obtain an infinite time. This is due to the exponential decay of $\mathrm{E}[X_t]$ through the term e^{-t}. However, we can circumvent this problem by recurring to the concept of half-life from exponential decaying radioactive processes (see e.g. Krane, 1987). The half-life is defined as the time taken until half of the atoms from a radionuclide have followed a decay process. In the same spirit, the following definition introduces the notion of δ-life.

Definition 8.2 (δ-life)**.** *Consider a stochastic process $X_t > 0$ whose expectation follows an exponential decay i.e. $dE[X_t]/dt = -aE[X_t]$ for $a \in \mathbb{R}^+$. The δ-life namely τ_δ is defined as the time when $E[X_t]/E[X_0] = \delta$ and has a value of*

$$\tau_\delta = \frac{\ln(1/\delta)}{a}.$$

As used in Section 8.4, we will focus on the number of 1-bits on linear functions, then we establish our satisfiability criterion as $\delta = 1/n$. This implies that in the worst case, where the system starts at a distance n from the long term expectation we will be satisfied when there is only a difference of one 1-bit.

Theorem 8.9. *The $1/n$-life of the Non-Elitist Multiplicative Drift (Theorem 8.6) is given by*

$$\tau_{1/n} \leq \frac{n \ln n}{p^{\uparrow}}.$$

Proof. The result for the expectation from Theorem 8.6 can be rewritten as $\mathrm{E}[X_t] - b = (\mathrm{E}[X_0] - b) \cdot e^{-at}$. It is straightforward to see that the process $\mathrm{E}[X_t] - b$ follows an exponential decay, then using Definition 8.2 for $\delta = 1/n$ leads to

$$\tau_{1/n} = \frac{\ln n}{a} = \frac{n \ln n}{p^{\uparrow} + p^{\downarrow}} \leq \frac{n \ln n}{p^{\uparrow}},$$

where we have introduced the a coefficient from Equation (8.16). □

The following table includes the $1/n$-lives ($\tau_{1/n}$) derived from the SDE method with the above expression and expected optimisation times $\mathrm{E}[T]$ from the literature. Before making the comparison let us remember that optimisation times refer to hitting the optimum whereas the $1/n$-life refers to approaching equilibrium. If the parameters of the algorithm allow the stationary distribution, to be close enough to the optimum this will be a fair comparison. This is the reason why the results for the RW highly differ in the table, a RW can not optimise ONEMAX in polynomial time but can reach its stationary distribution. On the other hand the results for the three other algorithms are in high concordance with the literature (up to smaller terms).

ONEMAX	**RLS**	**RW**	**MA**	**SSWM**
$p^{\uparrow}$	1	1	1	$\frac{1-e^{-2\beta}}{1-e^{-2N\beta}}$
$\tau_{1/n}$	$n\ln n$	$n\ln n$	$n\ln n$	$\frac{n\ln n}{p^{\uparrow}}$
$\mathrm{E}[T]$	$n\ln n + O(n)$	$\Omega(2^n)$	$n\ln n + O(n)$	$\frac{n\ln n}{p^{\uparrow}} + O(n)$
γ	-	-	$\geq \ln n$	$\geq \ln n$

Table 8.5 $\tau_{1/n}$ from Theorem 8.9 and expected optimisation times $\mathrm{E}[T]$ on ONEMAX (RLS-e.g. Oliveto and Yao (2011), RW-Garnier et al. (1999), MA-Jansen (2007) and SSWM-Theorem 5.1). Where $p^{\uparrow} = p_{\mathrm{acc}}(1)$, $\gamma = \alpha$ for MA and $\gamma = 2\beta(N-1)$ for SSWM.

8.5.4 Simulations

Another sanity-check that we have considered to validate Hypothesis 8.1 is to perform numerical simulations. We built several graphical representations where the experimental results are represented by: coloured dots for the expectation and a wrinkled shadowed region that includes one standard deviation around the expectation. On the other hand, theoretical results are represented by dashed black lines. All experiments have a problem size of $n = 100$ and the results were averaged over 100 independent runs. The parameters of the algorithms also remained constant for each figure with $\alpha = 1$ for MA and $(\beta, N) = (1, 2)$ for SSWM. For aesthetics reasons we have not included the variance estimations in the figures.

Firstly, the left graph from Figure 8.1 shows the application of the additive drift for LEADINGONES. We can observe that for RLS this approach is very precise until the last part of the optimisation process. Whereas for the (1+1) EA there is a large gap, nevertheless the method properly represents the evolution of the (1+1) EA's fitness on this problem. In our opinion, the discrepancy gap is mainly due to two reasons:

- choosing an additive drift approach for a process whose drift varies depending on the state
- pessimistic estimations for the additive constant b (see Theorem 8.4 and Applications 8.1 and 8.2).

This deviation from additivity can be seen at the end of the simulation when the experimental lines start curving, whereas the theoretical lines remain straight until they hit the optimum.

Secondly, the right graph from Figure 8.1 compares RLS and the (1+1) EA on the ONEMAX problem. The comparison between experiments and theory for RLS is excellent, even better than in LEADINGONES due to the absence of the boundary condition. Again for the (1+1) EA the theory collects the exponential decrease of the process. However there is still a gap which, in our opinion, is mainly due to equivalent reasons:

- using a multiplicative drift when the drift is not exactly multiplicative
- pessimistic estimations for the multiplicative constant b (see Theorem 8.5 and Application 8.4).

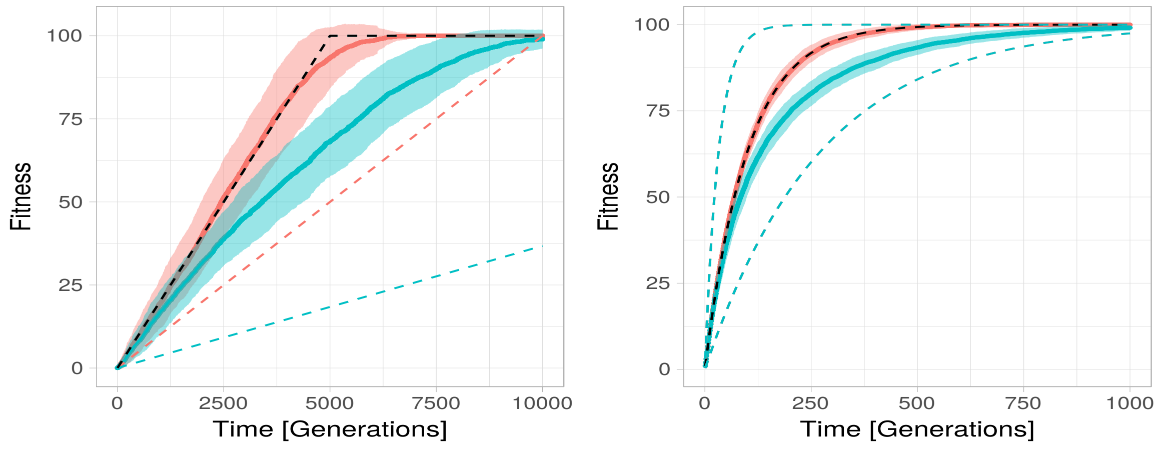


Fig. 8.1 Comparison of results for LEADINGONES (left) and ONEMAX (right). Solid lines represent the experimental results (RLS - red and (1+1) EA- blue). Dashed lines represent the theoretical results (RLS - black and (1+1) EA- blue). For LEADINGONES, the theoretical upper bound is shared for both algorithms and the lower bound for RLS is shown in red. Initialisation at 0^n. Problem size $n = 100$. Time budget 10000 (LEADINGONES) and 1000 (ONEMAX). Results were averaged over 100 runs and the shadowed zones include one standard deviation around the expectation.

The following figures show the application of the non-elitist drift for ONEMAX. The left graph from Figure 8.2 shows an excellent match between theoretical and experimental results for all the algorithms, describing the dynamics of these processes even when they are not able to optimise the problem. With the same accuracy the right graph of Figure 8.2 shows a case study where two algorithms (MA and RW) started in between their stationary expected value and the optimum but due to their weak selection strength they move away the optimum but towards the equilibrium's value.

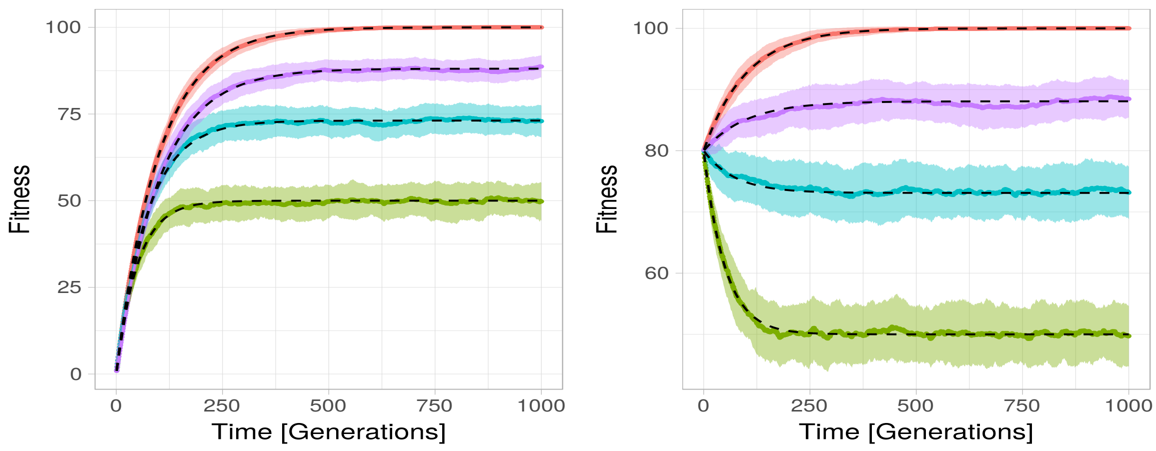


Fig. 8.2 Theoretical (dashed black lines) and experimental results (RLS - red, RW - green, MA - blue and SSWM - pink) on ONEMAX. Initialisation at 0^n (left) and $0^{20}1^{80}$ (right). Problem size $n = 100$. Time budget 1000. Algorithms' parameters: $\alpha = 1$ for MA and $(\beta, N) = (1, 2)$ for SSWM. Results were averaged over 100 runs and the shadowed zone includes one standard deviation around the expectation.

We would like to point out that our applications were just study cases to motivate the use of SDEs as an analysis tool for EAs. If we were aiming for precision we could have recycled already known and better bounds for the drift such us $b(Z_t) \leq -\frac{Z_t}{en}\left(1+\frac{16Z_t}{n}\right)$ from Doerr et al. (2011), but then it would become a variable drift process (Theorem 2.11).

8.5.5 Experimental Error

The previous sanity-check was based on a visual representation of experimental and theoretical results. However, graphical representations can sometimes be misleading and we have to be sure that the results obtained were not just an aesthetic effect. In order to objectively validate the SDE method (Hypothesis 8.1) we perform an experimental measurement of the error. We will use the widely used concepts of mean absolute percentage error (MAPE) and root mean squared error (RMSE) (see e.g. Hyndman and Koehler, 2006).

Definition 8.3. *Let $Y_{i=1,2,\ldots,n}$ be the vector of observed values and $\overline{Y}_{i=1,2,\ldots,n}$ the vector of predicted values. Then, the mean absolute percentage error (MAPE) and the root mean squared error (RMSE) are given by*

$$\mathrm{MAPE}(Y_i) = \frac{100\%}{n}\sum_{i=1}^{n}\left|\frac{\overline{Y_i}-Y_i}{Y_i}\right|, \qquad \mathrm{RMSE}(Y_i) = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(\overline{Y_i}-Y_i\right)^2}.$$

ONEMAX	MAPE(E $[X_t]$)	RMSE(Var $[X_t]$)
RLS	0.15%	0.41
(1+1) EA lower bound	16.34%	0.71
(1+1) EA upper bound	22.18%	1.30
RW	0.88%	0.42
MA	0.62%	0.44
SSWM	0.45%	0.37

Table 8.6 Experimental errors for the expectation and variance of the five studied algorithms on ONEMAX. The observed values are obtained from experiments. The predicted values are those derived in applications 8.3, 8.4 and 8.5. Algorithms' parameters: $\alpha = 1$ for MA and $(\beta, N) = (1, 2)$ for SSWM. Initialisation at 0^n. Problem size $n = 100$. Time budget 1000.

The advantage of the MAPE is that it takes into consideration the magnitude of the observed values. It produces a result in the form of a percentage that is interpreted as the magnitude of the deviation with reality. However, for small values of the observed values this measurement is not appropriate due to the singularity at $Y_i = 0$. This is a problem for measuring the error of the variance due to the deterministic initialisation used in the experiments and the concentration of the independent runs towards the same fitness value at the end of the simulation. For this reason, we have decided to use the MAPE for the expectations and the RMSE for the variance.

The following tables show agreement with Figures 8.1 and 8.2. The results for all the algorithms with local mutations, i.e. all but the (1+1) EA, are excellent on ONEMAX. With MAPE of the expectation being even smaller than 1%. Only for the (1+1) EA we can see a significant difference, due to the reasons explained in the previous section.

Finally, the comparison against LEADINGONES (Table 8.7) is not as successful as in the previous case. Here, the errors are quite large as shown in the left graph of Figure 8.1. But again, the reasons for the discrepancy are not intrinsic to using SDEs, they are due to the chosen drift technique.

8.6 Conclusions

For the last decade rigorous runtime analysis has become immensely popular, in part due to the emergence of the so-called "drift theorems" that inform about the time to reach a particular state or set of states. However, these results are not directly translatable for fixed

LEADINGONES	MAPE(E $[X_t]$)	RMSE(Var $[X_t]$)
RLS lower bound	47.90%	7.82
RLS upper bound	3.39%	7.73
(1+1) EA lower bound	72.27%	8.03
(1+1) EA upper bound	23.74%	7.93

Table 8.7 Experimental errors for the expectation and variance of RLS and the (1+1) EA on LEADINGONES. The observed values are obtained from experiments. The predicted values are those derived in applications 8.1 and 8.2. Initialisation at 0^n. Problem size $n = 100$. Time budget 10000.

budget scenarios which ask about the opposite question: how much improvement can be obtained in a fixed number of fitness evaluations. This problem is relevant in practice: in real problems it is usually impossible to know whether the optimum has been found. In this case, it is arguably more useful to know how much improvement an algorithm can achieve in a fixed number of steps.

Here, we have introduced the use of SDEs to the study of EAs which seem particularly suited to obtain results for fixed budget scenarios. Even though SDEs are approximations of the dynamics of EAs, they can produce analytical insight which is not possible with other tools, such as Markov chains. At the same time, SDEs do not discard the stochasticity of the dynamics and allow for the estimation of the variance of relevant quantities, such as the state at which the algorithm will find itself after t iterations.

Here we made a simple preliminary exploration of the potential of these tools for the study of EAs. We did not make use of other techniques for the study of SDEs that allow for greater insight into the dynamics of a stochastic process. For example, the Fokker-Planck equation associated with a SDE can be used to track the full probability distribution of the process. Even though this is, in many instances, impossible to solve analytically it can reveal what are the crucial parameters of a stochastic process. Furthermore, it is often solvable for the steady state, allowing for the characterisation of the stationary distribution of the process. In fact, these techniques are commonly used in the related field of population genetics where they have been used to provide foundational results. The results we presented here are in themselves interesting, especially when applied to fixed budget analysis, but they also hold the promise that SDEs can become a standard tool in EC.

We believe there is plenty of room for future work by considering more complex functions or more elaborated heuristics. But the main challenge would be a rigorous translation of the SDE method presented here.

Part IV

Conclusions and Outlook

Chapter 9

Conclusions

This PhD thesis has focused on bridging population genetics and evolutionary computation. Our work has substantially contributed to breaking down the walls between both communities, exploiting the unexplored intersection between both fields. We made use of the methods and techniques independently derived by PG and EC to study open problems from a new perspective.

By successfully quantifying the complexity of adaptive walks on many landscapes, we showed that theoretical biology can highly benefit from the tools used for the runtime analysis of evolutionary algorithms. In addition, EC also profits from this synergetic relationship. Although the theoretical study of non-elitist algorithms is still in its early days, it is a well studied characteristic of natural evolution. Apart from the direct gain from the runtime analysis of the SSWM algorithm, we used diffusion theory (one of the favourite tools amongst theoretical geneticists) to derive the first fixed budget result for non-elitist algorithms. Overall, we have derived plenty of meaningful results for both communities, showing that this interdisciplinary approach is effective and relevant. Proof of this is that our work has been published in high rated conferences and top journals from both fields (see Section 1.1).

In Chapter 1 we motivated not only the importance of studying both fields independently, but also their highly unexplored intersection. Then, we introduced the reader to the fields of Evolutionary Algorithms (Chapter 2) and Population Genetics (Chapter 3), specially to the mathematical techniques for the runtime analysis of randomised heuristics (Section 2.3) and to the derivation of the so-called Strong Selection Weak Mutation evolutionary regime (Subsection 3.2.3). In Chapter 4 we explained the importance of measuring the time that natural populations need to evolve complex adaptations. We described the state of the art for such analysis and how our proposed approach based on the runtime analysis of randomised algorithms could succeed.

In Chapter 5 we analysed hill-climbing problems. We saw that for additive fitness landscapes, it is sufficient that the selection strength is above a certain threshold for populations to be able to efficiently climb to the fitness peak. However, when considering a class of epistatic landscapes characterised by a single mutational path of strictly increasing fitness, we found that this limit no longer applies and that a constant selection strength will enable a population to climb to the optimum, albeit at a slower rate than in an additive landscapes. From a computational perspective, we showed that SSWM can take advantage of information about the steepest gradient, outperforming classical evolutionary algorithms such as the (1+1) EA.

In Chapter 6 we presented an analysis of randomised search heuristics for crossing fitness valleys where no mutational bias exists. Our focus was to highlight characteristics of valleys where an elitist selection strategy should be preferred to a non-elitist one and vice-versa. We rigorously proved that while the (1+1) EA is efficient for valleys and valley paths up to moderate lengths, both SSWM and MA are efficient when the valleys and valley paths are not too deep. Finally, we considered a version of SSWM that effectively combines non-elitism with global mutations to tunnel through sharp valleys.

In Chapter 7 we investigated SSWM and MA with the goal of identifying function characteristics where rejecting improvements is beneficial. We found a starting hurdle, both algorithms have the same stationary distribution. Hence, they may only have considerably different performance on optimisation functions where the mixing time is large. Our analysis on a 3 state model highlights that a simple function with a local optimum of low fitness and a global optimum of high fitness does not allow the required large mixing times. The reason is that, although MA initially chooses the local optimum more often than SSWM, it still escapes quickly. As a result we designed a 5 state model which "locks" the algorithms to their initial choices. By amplifying the function to contain several copies of the 5 state model we achieve our goal of defining a composite function where SSWM is efficient while MA requires exponential time with overwhelming probability, independent from its temperature parameter. Furthermore, we presented a further comparison with exploiting algorithms such as BILS, FILS and a (1,λ) RLS. While BILS excelled, experiments have shown that the (1,λ) RLS is efficient only for enough large values of λ.

Finally, we considered if the techniques from Population Genetics could be applied for the analysis of evolutionary algorithms. In Chapter 8, we introduced the use of stochastic differential equations to the study of evolutionary algorithms. This approach was based on the diffusion theory used by geneticists to track the change in the allele frequencies of a population (Section 3.2). Although we made a simple preliminary exploration of the potential of these tools for the study of EAs, we managed to derived several drift theorems for a fixed budget analysis. We claim that our results can be the starting point of a new tool for the theoretical analysis of randomised search heuristics.

As general overview, this thesis has yielded many contributions for both PG and EC. The main contribution of this thesis was the plethora of results derived for the SSWM algorithm. We showed that SSWM is not only relevant for biology but that, as an optimisation algorithm, it can outperform many optimisers such as the (1+1) EA or MA. Although our main focus was SSWM, we did not lose generality in our analysis. We worked using general frameworks that made our results applicable not only for SSWM but also for MA (Chapter 6) or even for a broader family of trajectory-based algorithms (Chapters 7 and 8). In addition, we have improved some of the already existing analysis tools like our extension of the Gambler's Ruin problem for non-elitist algorithms (Theorem 6.2 and Lemmas 6.6 and 6.7). Furthermore, in Chapter 8 we developed a new method for the analysis of randomised search heuristics within the new perspective of fixed-budget.

Future Work

Although we have a presented a substantial amount of novel results, there are still many open questions in biology where runtime analysis can yield interesting results. For example runtime analysis can pave the way for a rigorous translation of the highly debated Wright's shifting balance theory (Wright, 1932). As described in Chapter 6, this theory states that on fitness landscapes with multiple peaks where isolated populations might fail, parallel populations can cooperate to effectively adapt on such landscapes.

Another problem candidate is to analyse the SSWM regime on the stabilising selection model. Here each bit x_i is assigned a weight w_i and the mathematical problem is to minimise the distance $W - \sum_{i=1}^{n} x_i \cdot w_i$ to a predefined target W. It is straightforward to notice that this scenario can be related to some well known problems in EC (SUBSETSUM, KNAPSACK and PARTITION) from where there are already results for the (1+1) EA (Sudholt, 2011a; Witt, 2005; Zhou et al., 2007).

From a computational perspective, we would like to study other evolutionary regimes that correspond to more elaborated EAs. In particular, we think it would be interesting to include crossover and try to shed some light on the open question of the role of recombination in biology (Barton and Charlesworth, 1998). In addition, we have mainly considered the SSWM regime which is well described by a trajectory-based algorithm. However, other population-based evolutionary regimes might not be able to be cast as a trajectory algorithm. Hence an important direction for future work would be extending our analysis for population-based algorithms.

Last but not least, we think that the approach presented on Chapter 8 is very promising and has opened the door for many research directions. An extension for *parallel algorithms* seems natural since there exist work on systems of SDEs (see e.g. Øksendal, 2003). We also have not considered *dynamical problems*, however SDEs would allow to model time dependencies (recall that initially the drift $b(X_t,t)$ and diffusion $\sigma(X_t,t)$ coefficients were time dependent).

The method presented in Chapter 8 provides fixed-budget results, which, as argued in Section 2.4, can yield more insight on the optimisation process and it is closer to the experimental application than the traditional runtime analysis. In contrast with other fixed-budget approaches, our method allows to derive (with little extra cost) an expression for the variance. This way we not only know the expected behaviour of the algorithm but also how much the EA deviates from this expectation.

References

Aita, T., Uchiyama, H., Inaoka, T., Nakajima, M., Kokubo, T., and Husimi, Y. (2000). Analysis of a local fitness landscape with a model of the rough Mt. Fuji-type landscape: Application to prolyl endopeptidase and thermolysin. *Biopolymers*, 54(1):64–79.

Akimoto, Y., Auger, A., and Hansen, N. (2012). Convergence of the continuous time trajectories of isotropic evolution strategies on monotonic $\mathscr{C}^2$-composite functions. In *Parallel Problem Solving from Nature - PPSN XII: 12th International Conference, Taormina, Italy, September 1-5, 2012, Proceedings, Part I*, pages 42–51. Springer Berlin Heidelberg.

Aldous, D. and Fill, J. (2017). *Reversible Markov Chains and Random Walks on Graphs*. Monograph in preparation.

Altenberg, L. (1994). The schema theorem and price's theorem. In *Proceedings of the Third Workshop on Foundations of Genetic Algorithms. Estes Park, Colorado, USA, July 31 - August 2 1994*, pages 23–49.

Auger, A. and Doerr, B., editors (2011). *Theory of Randomized Search Heuristics*. World Scientific.

Barnard, C. and Simon, H. A. (1947). *Administrative behavior. A study of decision-making processes in administrative organization*. Macmillan, New York.

Barton, N., Briggs, D. E. G., Eisen, J., Goldstein, D., and Patel, N. (2007). *Evolution*. Cold Spring Harbor Laboratory Press.

Barton, N. H. and Charlesworth, B. (1998). Why Sex and Recombination? *Science*, 281(5385):1986–1990.

Beyer, H. (2001). *The Theory of Evolution Strategies*. Natural Computing Series. Springer.

Black, F. and Scholes, M. (1973). The pricing of options and corporate liabilities. *Journal of Political Economy*, 81(3):637–654.

Boese, K. D., Kahng, A. B., and Muddu, S. (1994). A new adaptive multi-start technique for combinatorial global optimizations. *Operations Research Letters*, 16(2):101–113.

Chastain, E., Livnat, A., Papadimitriou, C., and Vazirani, U. (2014). Algorithms, games, and evolution. *Proceedings of the National Academy of Sciences*, 111(29):10620–10623.

Chatterjee, K., Pavlogiannis, A., Adlam, B., and Nowak, M. A. (2014). The time scale of evolutionary innovation. *PLoS Computational Biology*, 10(9).

Chen, G.-Y. and Saloff-Coste, L. (2013). On the mixing time and spectral gap for birth and death chains. *Latin American Journal of Probability and Mathematical Statistics*, X:293–321.

Cormen, T., Leiserson, C., Rivest, R., and Stein, C. (2001). *Introduction To Algorithms*. MIT Press.

Corus, D., Dang, D.-C., Eremeev, A. V., and Lehre, P. K. (2014). Level-based analysis of genetic algorithms and other search processes. In *Parallel Problem Solving from Nature (PPSN)*, pages 912–921. Springer.

Corus, D., He, J., Jansen, T., Oliveto, P. S., Sudholt, D., and Zarges, C. (2017). On easiest functions for mutation operators in bio-inspired optimisation. *Algorithmica*, 78(2):714–740.

Cox, J. C., Ingersoll, J. E., and Ross, S. A. (1985). A Theory of the Term Structure of Interest Rates. *Econometrica*, 53(2):385–407.

Coyne, J. A., Barton, N. H., and Turelli, M. (2000). Is wright's shifting balance process important in evolution? *Evolution*, 54(1):306–317.

Crow, J. and Kimura, M. (2009). *An Introduction to Population Genetics Theory*. Blackburn Press.

Dang, D.-C., Friedrich, T., Kötzing, T., Krejca, M. S., Lehre, P. K., Oliveto, P. S., Sudholt, D., and Sutton, A. M. (2017). Escaping local optima using crossover with emergent diversity. *IEEE Transactions on Evolutionary Computation*. To appear.

Darwin, C. (1859). *The Origin of Species*. John Murray.

Desai, M. M., Fisher, D. S., and Murray, A. W. (2007). The Speed of Evolution and Maintenance of Variation in Asexual Populations. *Current Biology*, 17(5):385–394.

Doerr, B. (2011). Analyzing randomized search heuristics: Tools from probability theory. In *Auger and Doerr (2011)*, pages 1–20. World Scientific.

Doerr, B., Doerr, C., and Yang, J. (2016). Optimal parameter choices via precise black-box analysis. In *Proceedings of the Genetic and Evolutionary Computation Conference 2016*, GECCO '16, pages 1123–1130, New York, NY, USA. ACM.

Doerr, B., Fouz, M., and Witt, C. (2011). Sharp bounds by probability-generating functions and variable drift. In *Proceedings of the 13th Annual Conference on Genetic and Evolutionary Computation*, GECCO '11, pages 2083–2090, New York, NY, USA. ACM.

Doerr, B. and Goldberg, L. A. (2010). Drift analysis with tail bounds. In *Proceedings of the 11th International Conference on Parallel Problem Solving from Nature: Part I*, PPSN'10, pages 174–183, Berlin, Heidelberg. Springer-Verlag.

Doerr, B., Jansen, T., Witt, C., and Zarges, C. (2013). A method to derive fixed budget results from expected optimisation times. In *Proceeding of the fifteenth annual conference on Genetic and evolutionary computation (GECCO '13)*, pages 1581–1588. ACM.

Doerr, B., Johannsen, D., and Winzen, C. (2012). Multiplicative drift analysis. *Algorithmica*, 64(4):673–697.

Droste, S., Jansen, T., and Wegener, I. (2002). On the analysis of the (1+1) evolutionary algorithm. *Theoretical Computer Science*, 276:51–81.

Droste, S., Jansen, T., and Wegener, I. (2006). Upper and lower bounds for randomized search heuristics in black-box optimization. *Theory Computing Systems*, 39(4):525–544.

Eiben, A. E. and Smith, J. E. (2015). *Introduction to Evolutionary Computing*. Springer, 2nd edition.

Einstein, A. (1905). Über die von der molekularkinetischen Theorie der Wärme geforderte Bewegung von in ruhenden Flüssigkeiten suspendierten Teilchen. *Annalen der Physik*, 322(8):549–560.

Euler, L. (1744). *Methodus inveniendi lineas curvas maximi minimive proprietate gaudentes, sive solutio problematis isoperimetrici lattissimo sensu accepti*. apud Marcum-Michaelem Bousquet et socios.

Ewens, W. (2004). *Mathematical Population Genetics 1: Theoretical Introduction*. Interdisciplinary Applied Mathematics. Springer New York.

Eyre-Walker, A. and Keightley, P. D. (2007). The distribution of fitness effects of new mutations. *Nature Review Genetics*, 8(8):610–618.

Feller, W. (1949). On the theory of stochastic processes, with particular reference to applications. In *Proceedings of the [First] Berkeley Symposium on Mathematical Statistics and Probability*, pages 403–432, Berkeley, Calif. University of California Press.

Feller, W. (1968). *An introduction to probability theory and its applications*. Wiley.

Fisher, R. A. (1930). *The genetical theory of natural selection.* Oxford Clarendon Press.

Fogel, L. J., Owens, A. L., and Walsh, M. L. (1965). Artificial intelligence through simulation of evolution. In *Biophysics and Cybernetic Systems*, pages 131–156. Spartan, Washington.

Fogle, C. A., Nagle, J. L., and Desai, M. M. (2008). Clonal Interference, Multiple Mutations and Adaptation in Large Asexual Populations. *Genetics*, 180(4):2163–2173.

Friedrich, T., Kötzing, T., and Wagner, M. (2017). A generic bet-and-run strategy for speeding up stochastic local search. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, February 4-9, 2017, San Francisco, California, USA.*, pages 801–807.

Garnier, J., Kallel, L., and Schoenauer, M. (1999). Rigorous hitting times for binary mutations.

Gerrish, P. and Lenski, R. (1998). The fate of competing beneficial mutations in an asexual population. *Genetica*, 102-103:127–144.

Gießen, C. (2013). Hybridizing evolutionary algorithms with opportunistic local search. In *Proceedings of the 15th Annual Conference on Genetic and Evolutionary Computation (GECCO '13)*, pages 797–804. ACM.

Gillespie, J. H. (1983). Some properties of finite populations experiencing strong selection and weak mutation. *The American Naturalist*, 121(5):691–708.

Gillespie, J. H. (1984). Molecular evolution over the mutational landscape. *Evolution*, 38(5):1116–1129.

Gillespie, J. H. (1989). *When Not to Use Diffusion Processes in Population Genetics*, pages 57–70. Princeton University Press.

Goldberg, D. (1987). Simple genetic algorithms and the minimal, deceptive problem. In Davis, L., editor, *Genetic algorithms and simulated annealing*, pages 74–88. Pitman.

Goodnight, C. J. and Wade, M. J. (2000). The ongoing synthesis: a reply to coyne, barton, and turelli. *Evolution*, 54(1):317–324.

Grafen, A. (2014). The formal darwinism project in outline. *Biology & Philosophy*, 29(2):155.

Grant, V. and Flake, R. H. (1974). Solutions to the Cost-of-Selection Dilemma. *Proceedings of the National Academy of Sciences of the United States of America*, 71(10):3863–3865.

Hajek, B. (1982). Hitting-time and occupation-time bounds implied by drift analysis with applications. *Advances in Applied Probability*, 14(3):502–525.

Haldane, J. B. S. (1957). The cost of natural selection. *J. Genet.*, 55:511–524.

Haldane, J. B. S. (1990). A mathematical theory of natural and artificial selection—i. *Bulletin of Mathematical Biology*, 52(1):209–240.

Hardy, G. H. (1908). Mendelian proportions in a mixed population. *Journal of Biology and Medicine*, 2(76):79—80.

He, J. and Yao, X. (2001). Drift analysis and average time complexity of evolutionary algorithms. *Artificial Intelligence*, 127:57–85.

He, J. and Yao, X. (2003). Towards an analytic framework for analysing the computation time of evolutionary algorithms. *Artificial Intelligence*, 145(1):59 – 97.

Holland, J. H. (1973). Genetic algorithms and the optimal allocation of trials. *SIAM Jounral on Computing*, 2:88–105.

Holland, J. H. (1975). *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control and Artificial Intelligence*. MIT Press, Cambridge, MA, USA. Reprint edition 1992 (originally published in 1975).

Horn, J., Goldberg, D. E., and Deb, K. (1994). Long path problems. In *Parallel Problem Solving from Nature (PPSN III)*, volume 866 of *LNCS*, pages 149–158.

Hyndman, R. J. and Koehler, A. B. (2006). Another look at measures of forecast accuracy. *International Journal of Forecasting*, 22(4):679 – 688.

Itô, K. (1944). Stochastic integral. In *Proceedings of the Imperial Academy*.

Itô, K. (1946). On a stochastic integral equation. In *Proceedings of the Japan Academy*.

Itô, K. (1950). Stochastic differential equations in a differentiable manifold. *Nagoya Mathematical Journal*.

Itô, K. (1951). Diffusion processes and their sample paths. *Nagoya Mathematical Journal*.

Jägersküpper, J. and Storch, T. (2007). When the plus strategy outperforms the comma strategyand when not. In *2007 IEEE Symposium on Foundations of Computational Intelligence*, pages 25–32.

Jansen, T. (2007). On the brittleness of evolutionary algorithms. In *Foundations of Genetic Algorithms: 9th International Workshop, FOGA 2007, Mexico City, Mexico, January 8-11, 2007, Revised Selected Papers*, pages 54–69, Berlin, Heidelberg. Springer Berlin Heidelberg.

Jansen, T. (2013). *Analyzing Evolutionary Algorithms*. Springer-Verlag Berlin Heidelberg.

Jansen, T. and Wegener, I. (2002). The analysis of evolutionary algorithms - A proof that crossover really can help. *Algorithmica*, 34(1):47–66.

Jansen, T. and Wegener, I. (2007). A comparison of simulated annealing with a simple evolutionary algorithm on pseudo-Boolean functions of unitation. *Theoretical Computer Science*, 386(1-2):73–93.

Jansen, T. and Zarges, C. (2012). Fixed budget computations: a different perspective on run time analysis. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO '12)*, pages 1325–1332. ACM.

Jerrum, M. and Sinclair, A. (1996). The Markov chain Monte Carlo method: an approach to approximate counting and integration. In *Approximation Algorithms for NP-hard Problems*, pages 482–520. PWS Publishing.

Johannsen, D. (2010). *Random Combinatorial Structures and Randomized Search Heuristics*. PhD thesis, Universität des Saarlandes, Saarbrücken, Germany and the Max-Planck-Institut für Informatik.

Karlin, S. and McGregor, J. (1964). *On some stochastic model in population genetics*, pages 245–278. University of Wisconsin Press.

Karlin, S. and Taylor, H. M. (1981). *A Second Course in Stochastic Processes*. Academic Press, New York, 1 edition edition.

Kauffman, S. and Levin, S. (1987). Towards a general theory of adaptive walks on rugged landscapes. *Journal of Theoretical Biology*, 128(1):11–45.

Kauffman, S. A. and Weinberger, E. D. (1989). The NK model of rugged fitness landscapes and its application to maturation of the immune response. *Journal Theoretical Biology*, 141(2):211–245.

Kim, Y. and Orr, H. A. (2005). Adaptation in sexuals vs. asexuals: clonal interference and the Fisher-Muller model. *Genetics*, 171(3):1377–86.

Kimura, M. (1957). Some Problems of Stochastic Processes in Genetics. *The Annals of Mathematical Statistics*, 28(4):882–901.

Kimura, M. (1961). Natural selection as the process of accumulating genetic information in adaptive evolution. *Genetics Research*, 2(01):127–140.

Kimura, M. (1962). On the probability of fixation of mutant genes in a population. *Genetics*, 47(6):713–719.

Kimura, M. (1964). Diffusion Models in Population Genetics. *Journal of Applied Probabilitiy*, 1(2):177–232.

Kimura, M. (1968). Genetic Variability Maintained in a Finite Population Due to Mutational Production of Neutral and Nearly Neutral Isoalleles. *Genetics Research*, 11(03):247–270.

Kimura, M. (1983). *The Neutral Theory of Molecular Evolution*. Cambridge University Press.

Kingman, J. F. C. (1978). A Simple Model for the Balance between Selection and Mutation. *Journal of Applied Probability*, 15(1):1–12.

Kondrashov, F. A. and Kondrashov, A. S. (2001). Multidimensional epistasis and the disadvantage of sex. *Proceedings of the National Academy of Sciences of the United States of America*, 98(21):12089–92.

Koza, J. R. (1992). *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. MIT Press, Cambridge, MA, USA.

Krane, K. S. (1987). *Introductory nuclear physics*. Wiley, New York.

Kryazhimskiy, S., Tkacik, G., and Plotkin, J. B. (2009). The dynamics of adaptation on correlated fitness landscapes. *Proceedings of the National Academy of Sciences of the United States of America*, 106(44):18638–18643.

Lamarck, J.-B. (1809). *Philosophie Zoologique*. Museum d'Histoire Naturelle.

Lehre, P. K. and Witt, C. (2012a). Black-box search by unbiased variation. *Algorithmica*, 64(4):623–642.

Lehre, P. K. and Witt, C. (2012b). Black-box search by unbiased variation. *Algorithmica*, 64(4):623–642.

Lengler, J. and Spooner, N. (2015). Fixed budget performance of the (1+1) EA on linear functions. In *Proceedings of the 2015 ACM Conference on Foundations of Genetic Algorithms XIII*, FOGA '15, pages 52–61, New York, NY, USA. ACM.

Levin, D. A., Peres, Y., and Wilmer, E. L. (2008). *Markov Chains and Mixing Times*. American Mathematical Society.

Luque, G. and Alba, E. (2011). *Parallel Genetic Algorithms. Theory and Real World Applications*. Springer-Verlag.

Maghsoodi, Y. (1996). Solution of the Extended Cir Term Structure and Bond Option Valuation. *Mathematical Finance*, 6(1):89–109.

Malthus, T. R. (1798). *An Essay on the Principle of Population*. J. Johnson, London.

Mendel, G. (1866). Experiments on plant hybridization. *Proceedings of the Natural History Society of Brünn*.

Merz, P. and Freisleben, B. (1998). Memetic algorithms and the fitness landscape of the graph bi-partitioning problem. In *Parallel Problem Solving from Nature (PPSN V)*, pages 765–774. Springer Berlin Heidelberg.

Metropolis, N., Rosenbluth, A. W., Rosenbluth, M. N., Teller, A. H., and Teller, E. (1953). Equation of state calculations by fast computing machines. *The Journal of Chemical Physics*, 21(6):1087–1092.

Mitzenmacher, M. and Upfal, E. (2005). *Probability and Computing: Randomized Algorithms and Probabilistic Analysis*. Cambridge University Press.

Moran, P. (1963). On the nonexistence of adaptive topographies. *Annals of human genetics*, 27(4):383–393.

Motwani, R. and Raghavan, P. (1995). *Randomized Algorithms*. Cambridge International Series on Parallel Computation. Cambridge University Press.

Mühlenbein, H. (2009). Evolutionary computation: Centralized, parallel or collaborative. *Computational Intelligence*, pages 561–595.

Muhlenbein, H. and Mahnig, T. (2002). Evolutionary computation and wright's equation. *Theoretical Computer Science*, 287(1):145–165.

Mühlenbein, H. and Schlierkamp-Voosen, D. (1993). Predictive models for the breeder genetic algorithm i. continuous parameter optimization. *Evolutionary computation*, 1(1):25–49.

Nallaperuma, S., Neumann, F., and Sudholt, D. (2017a). Expected fitness gains of randomized search heuristics for the traveling salesperson problem. *Evolutionary Computation*. To appear.

Nallaperuma, S., Oliveto, P. S., Pérez Heredia, J., and Sudholt, D. (2017b). On the analysis of trajectory-based search algorithms: When is it beneficial to reject improvements or to exploit? *Algorithmica*. Submitted to.

Nallaperuma, S., Oliveto, P. S., Pérez Heredia, J., and Sudholt, D. (2017c). When is it beneficial to reject improvements? In *Proceedings of the Genetic and Evolutionary Computation Conference*, GECCO '17, pages 1391–1398, New York, NY, USA. ACM.

Neumann, F., Oliveto, P. S., and Witt, C. (2009). Theoretical analysis of fitness-proportional selection: landscapes and efficiency. In *Proceedings of the 2009 Genetic and Evolutionary Computation Conference (GECCO '09)*, pages 835–842. ACM Press.

Neumann, F. and Witt, C. (2010). *Bioinspired Computation in Combinatorial optimization*. Springer.

Nowak, M. A. (2006). *Evolutionary Dynamics*. Harvard University Press.

Ochoa, G. and Veerapen, N. (2016). Deconstructing the big valley search space hypothesis. In *Proceedings of the 16th European Conference on Evolutionary Computation in Combinatorial Optimization (EvoCOP 2016)*, pages 58–73. Springer International Publishing.

Øksendal, B. (2003). *Stochastic Differential Equations: An Introduction with Applications*. University of Michigan Press.

Oliveto, P. S., Lehre, P. K., and Neumann, F. (2009). Theoretical analysis of rank-based mutation - combining exploration and exploitation. In *Proceedings of the 2009 IEEE Congress on Evolutionary Computation (CEC '09)*, pages 1455–1462. IEEE Press.

Oliveto, P. S., Paixão, T., Pérez Heredia, J., Sudholt, D., and Trubenová, B. (2016). When non-elitism outperforms elitism for crossing fitness valleys. In *Proceedings of the Genetic and Evolutionary Computation Conference 2016*, GECCO '16, pages 1163–1170, New York, NY, USA. ACM.

Oliveto, P. S., Paixão, T., Pérez Heredia, J., Sudholt, D., and Trubenová, B. (2017). How to escape local optima in black box optimisation: When non-elitism outperforms elitism. *Algorithmica*. To appear.

Oliveto, P. S. and Sudholt, D. (2014). On the runtime analysis of stochastic ageing mechanisms. In *Proceedings of the 2014 Genetic and Evolutionary Computation Conference (GECCO '14)*, pages 113–120. ACM Press.

Oliveto, P. S. and Witt, C. (2008). Simplified drift analysis for proving lower bounds in evolutionary computation. In *Proceedings of the 10th International Conference on Parallel Problem Solving from Nature: PPSN X*, pages 82–91, Berlin, Heidelberg. Springer-Verlag.

Oliveto, P. S. and Witt, C. (2011). Simplified drift analysis for proving lower bounds in evolutionary computation. *Algorithmica*, 59(3):369–386.

Oliveto, P. S. and Witt, C. (2012). Erratum: Simplified drift analysis for proving lower bounds in evolutionary computation. *ArXiv e-prints*.

Oliveto, P. S. and Witt, C. (2014). On the runtime analysis of the simple genetic algorithm. *Theoretical Computer Science*, 545:2–19.

Oliveto, P. S. and Yao, X. (2011). Runtime analysis of evolutionary algorithms for discrete optimization. In *Auger and Doerr (2011)*, Series on Theoretical Computer Science. World Scientific.

Orr, H. A. (1995). The population genetics of speciation: the evolution of hybrid incompatibilities. *Genetics*, 139:1805–1813.

Orr, H. A. (2000). The rate of adaptation in asexuals. *Genetics*, 155(2):961–968.

Orr, H. A. (2002). The population genetics of adaptation: The adaptation of DNA sequences. *Evolution*, 56(7):1317–1330.

Orr, H. A. (2005). The genetic theory of adaptation: a brief history. *Nature Review Genetics*, 6(2):119–127.

Orr, H. A. (2006). The Population Genetics of Adaptation on Correlated Fitness Landscapes: The Block Model. *Evolution*, 60(6):1113–1124.

Paixão, T., Badkobeh, G., Barton, N., Corus, D., Dang, D.-C., Friedrich, T., Lehre, P. K., Sudholt, D., Sutton, A. M., and Trubenová, B. (2015). Toward a unifying framework for evolutionary processes. *Journal of Theoretical Biology*, 383:28–43.

Paixão, T., Pérez Heredia, J., Sudholt, D., and Trubenová, B. (2015). First steps towards a runtime comparison of natural and artificial evolution. In *Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation*, GECCO '15, pages 1455–1462, New York, NY, USA. ACM.

Paixão, T. and Pérez Heredia, J. (2017). An application of stochastic differential equations to evolutionary algorithms. In *Proceedings of the 14th ACM/SIGEVO Conference on Foundations of Genetic Algorithms*, FOGA '17, pages 3–11, New York, NY, USA. ACM.

Paixão, T., Pérez Heredia, J., Sudholt, D., and Trubenová, B. (2017). Towards a runtime comparison of natural and artificial evolution. *Algorithmica*, 78(2):681–713.

Park, S.-C., Neidhart, J., and Krug, J. (2016). Greedy adaptive walks on a correlated fitness landscape. *Journal of Theoretical Biology*, 397:89–102.

Pérez Heredia, J. (2017). Modelling evolutionary algorithms with stochastic differential equations. *Evolutionary Computation*. To appear.

Pérez Heredia, J., Trubenová, B., Sudholt, D., and Paixão, T. (2017). Selection limits to adaptive walks on correlated landscapes. *Genetics*, 205(2):803–825.

Phillips, P. C. (2008). Epistasis–the essential role of gene interactions in the structure and evolution of genetic systems. *Nature Review Genetics*, 9(11):855–67.

Poelwijk, F. J., Kiviet, D. J., Weinreich, D. M., and Tans, S. J. (2007). Empirical fitness landscapes reveal accessible evolutionary paths. *Nature*, 445(7126):383–6.

Prügel-Bennett, A. (1997). Modelling evolving populations. *Journal of Theoretical Biology*, 185(1):81–95.

Prügel-Bennett, A. and Shapiro, J. L. (1994). Analysis of genetic algorithms using statistical mechanics. *Physical Review Letters*, 72(9):1305.

Rechenberg, I. (1973). *Evolutionsstrategie: optimierung technischer systeme nach prinzipien der biologischen evolution*. Frommann-Holzboog.

Reeves, C. (1999). Landscapes, operators and heuristic search. *Annals of Operations Research*, 86(0):473–490.

Rice, S. (2004). *Evolutionary Theory: Mathematical and Conceptual Foundations*. Sinauer.

Rohlfshagen, P., Lehre, P. K., and Yao, X. (2009). Dynamic evolutionary optimisation: an analysis of frequency and magnitude of change. In *Proceedings of the 2009 Genetic and Evolutionary Computation Conference (GECCO '09)*, pages 1713–1720. ACM Press.

Ross, S. (1996). *Stochastic Processes*. John Wiley & Sons.

Rowe, J. E. and Sudholt, D. (2014). The choice of the offspring population size in the $(1, \lambda)$ evolutionary algorithm. *Theoretical Computer Science*, 545:20–38.

Rudolph, G. (1997a). *Convergence Properties of Evolutionary Algorithms*. Verlag Dr. Kovač.

Rudolph, G. (1997b). How mutation and selection solve long-path problems in polynomial expected time. *Evolutionary Computation*, 4(2):195–205.

Rudolph, G. (1998). Finite markov chain results in evolutionary computation: A tour d'horizon. *Fundamenta Informaticae*, 35(1-4):67–89.

Sarker, R., Mohammadian, M., and Yao, X. (2002). *Evolutionary optimization*, volume 48. Springer Science & Business Media.

Sasaki, G. H. and Hajek, B. (1988). The time complexity of maximum matching by simulated annealing. *Journal of the ACM*, 35:387–403.

Schaul, T. (2012). Natural evolution strategies converge on sphere functions. In *Proceedings of the 14th Annual Conference on Genetic and Evolutionary Computation*, GECCO '12, pages 329–336, New York, NY, USA. ACM.

Schwefel, H.-P. P. (1993). *Evolution and Optimum Seeking: The Sixth Generation*. John Wiley & Sons, Inc., New York, NY, USA.

Sella, G. and Hirsh, A. E. (2005). The application of statistical physics to evolutionary biology. *Proceedings of the National Academy of Sciences of the United States of America*, 102(27):9541–9546.

Smith, J. E. (2007). Coevolving memetic algorithms: A review and progress report. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 37(1):6–17.

Stieltjes, T.-J. (1894). Recherches sur les fractions continues. *Annales de la Faculté des sciences de Toulouse : Mathématiques*, 8(4):J1–J122.

Stratonovich, R. (1966). A new representation for stochastic integrals and equations. *SIAM Journal on Control*.

Sudholt, D. (2008). *Computational complexity of evolutionary algorithms, hybridizations, and swarm intelligence*. PhD thesis.

Sudholt, D. (2009). The impact of parametrization in memetic evolutionary algorithms. *Theoretical Computer Science*, 410(26):2511–2528.

Sudholt, D. (2011a). Hybridizing evolutionary algorithms with variable-depth search to overcome local optima. *Algorithmica*, 59(3):343–368.

Sudholt, D. (2011b). Using markov-chain mixing time estimates for the analysis of ant colony optimization. In *Proceedings of the 11th Workshop Proceedings on Foundations of Genetic Algorithms*, FOGA '11, pages 139–150, New York, NY, USA. ACM.

Sudholt, D. (2013). A new method for lower bounds on the running time of evolutionary algorithms. *IEEE Transactions on Evolutionary Computation*, 17(3):418–435.

Turing, A. (1948). Intelligent machinery. In *Collected Works of A.M. Turing: Mechanical Intelligence*. Elsevier Science, 1992.

Unckless, R. L. and Orr, H. A. (2009). Dobzhansky-muller incompatibilities and adaptation to a shared environment. *Heredity*, 102(3):214–217.

Valiant, L. (2013). *Probably Approximately Correct: Nature's Algorithms for Learning and Prospering in a Complex World*. Basic Books, New York, first edition.

Valiant, L. G. (2009). Evolvability. *Journal of the ACM*, 56(1):3:1–3:21.

Vose, M. D. (1995). Modeling simple genetic algorithms. *Evolutionary Computation*, 3(4):453–472.

Wallace, A. R. (1855). On the law which has regulated the introduction of new species. *Annals And Magazine of Natural History*, 16:184–196.

Wegener, I. (2003). Methods for the analysis of evolutionary algorithms on pseudo-boolean functions. In Sarker, R., Mohammadian, M., and Yao, X., editors, *Evolutionary Optimization*, volume 48 of *International Series in Operations Research & Management Science*, chapter 14, pages 349–369. Kluwer Academic Publishers.

Wegener, I. (2005). Simulated annealing beats Metropolis in combinatorial optimization. In *Proceedings of the 32nd International Colloquium on Automata, Languages and Programming (ICALP '05)*, volume 3580 of *LNCS*, pages 589–601.

Wei, K. and Dinneen, M. J. (2014). Runtime analysis to compare best-improvement and first-improvement in memetic algorithms. In *Proceedings of the 2014 Annual Conference on Genetic and Evolutionary Computation (GECCO '14)*, pages 1439–1446. ACM.

Weinberg, W. (1908). Über den Nachweis der Vererbung beim Menschen. *Jahreshefte des Vereins für Vaterländische Naturkunde in Württemberg*, (64):369—382.

Weinreich, D. M., Watson, R. A., and Chao, L. (2005). Perspective: sign epistasis and genetic costraint on evolutionary trajectories. *Evolution*, 59(6):1165–1174.

Whitlock, M. C., Phillips, P. C., Moore, F. B.-G., and Tonsor, S. J. (1995). Multiple Fitness Peaks and Epistasis. *Annual Review of Ecology and Systematics*, 26:601–629.

Wilke, C. O. (2004). The Speed of Adaptation in Large Asexual Populations. *Genetics*, 167(4):2045–2053.

Witt, C. (2005). Worst-case and average-case approximations by simple randomized search heuristics. In *STACS 2005, 22nd Annual Symposium on Theoretical Aspects of Computer Science, Stuttgart, Germany, February 24-26, 2005, Proceedings*, pages 44–56.

Witt, C. (2008). Population size versus runtime of a simple evolutionary algorithm. *Theoretical Computer Science*, 403(1):104–120.

Wormald, N. C. (1995). Differential equations for random processes and random graphs. *Ann. Appl. Probab.*, 5(4):1217–1235.

Wright, A. H. and Rowe, J. E. (2001). Continuous dynamical system models of steady-state genetic algorithms. In Martin, W. N. and Spears, W. M., editors, *Foundations of Genetic Algorithms 6*, pages 209 – 225. Morgan Kaufmann, San Francisco.

Wright, S. (1932). The roles of mutation, inbreeding, crossbreeding and selection in evolution. *Proceedings of the Sixth International Congress on Genetics*, 1(6):356–366.

Wright, S. (1937). The distribution of gene frequencies in populations. *Proceedings of the National Academy of Sciences of the United States of America*, 23(6):307—320.

Wright, S. (1945). The differential equation of the distribution of gene frequencies. *Proceedings of the National Academy of Sciences of the United States of America*, 31(12):382—389.

Yin, G., Rudolph, G., and Schwefel, H.-P. (1995). Analyzing $(1,\lambda)$ evolution strategy via stochastic approximation methods. *Evolutionary Computation*, pages 473–489.

Zhou, Y., Guo, Z., and He, J. (2007). On the running time analysis of the (1+1) evolutionary algorithm or the subset sum problem. In *LSMS (1)*, volume 4688 of *Lecture Notes in Computer Science*, pages 73–82. Springer.

Appendix A

Probability Theory

We assume that the reader is familiar with the basic concepts of probability theory. However, for the sake of completion, we formally introduce in this appendix the concepts used within this thesis. Some of the statements are freely adapted from Chapter 1 from the textbook by Ross (1996) and Chapter 2 from the textbook by Mitzenmacher and Upfal (2005).

Definition A.1 (Expectation)**.** *Let X be a discrete random variable that takes values $x_1, x_2, \ldots$. Then, we define the expectation as*

$$E[X] = \sum_{i=1}^{\infty} x_i \cdot \Pr(X = x_i).$$

Definition A.2 (Conditional Expectation)**.** *Let X and Y be two random variables, we define the conditional expectation as*

$$E[X \mid Y = y] = \sum_{x} x \cdot \Pr(X = x \mid Y = y).$$

Lemma A.1 (Linearity of Expectation)**.** *Let X and Y be two random variables it holds that*

$$E[X + Y] = E[X] + E[Y].$$

Lemma A.2 (Law of Total Expectation)**.** *Let X and Y be two random variables it holds that*

$$E[X] = E[E[X \mid Y]].$$

Definition A.3 (Bernoulli Trial)**.** *A Bernoulli trial is a random process whose outcome has only two options: success and failure.*

Definition A.4 (Geometric Distribution)**.** *Let p be the success probability of a Bernoulli trial. A geometric distribution counts the number of trials X needed to obtain one success*

$$\Pr(X = k) = (1-p)^{k-1} p.$$

Lemma A.3. *The expectation of a geometric random variable with parameter p is $1/p$.*

Lemma A.4 (Chernoff Bounds)**.** *Let $X = X_1 + \ldots + X_n$ be the sum of independent random variables with $X_i \in \{0,1\}$ for all $1 \le i \le n$. Then*

$$\begin{aligned}
&\Pr(X \ge (1+\delta)E[X]) < \left(\frac{e^{\delta}}{(1+\delta)^{1+\delta}}\right)^{E[X]} \quad \textit{for } \delta > 0 \\
&\Pr(X \le (1-\delta)E[X]) \le \left(\frac{e^{-\delta}}{(1-\delta)^{1-\delta}}\right)^{E[X]} \quad \textit{for } 0 < \delta < 1 \\
&\Pr(X \ge (1+\delta)E[X]) \le e^{-E[X]\delta^2/3} \quad \textit{for } 0 < \delta < 1 \\
&\Pr(X \le (1-\delta)E[X]) \le e^{-E[X]\delta^2/2} \quad \textit{for } 0 < \delta < 1.
\end{aligned}$$

Appendix B

Diffusion Theory

This appendix contains the derivations of the diffusion equation omitted from Sections 3.2 and 8.2.

B.1 Kolmogorov Forward Equation

The time evolution of the probability density $p(x,t)$ of any stochastic process X_t can be described by the Chapman-Kolmogorov equation (see e.g. Feller, 1949)

$$p(x,t+\Delta t) = \int \Delta(\delta \mid x) \cdot p(x-\delta,t) \cdot d\delta, \tag{B.1}$$

where $\Delta(\delta \mid x)$ is the transition probability of reaching x from $x-\delta$. Let us consider the left-hand term: after performing a first order Taylor expansion around $\Delta t = 0$ we obtain

$$p(x,t+\Delta t) \approx p(x,t) + \Delta t \cdot \frac{\partial p(x,t)}{\partial t} \tag{B.2}$$

and by Taylor's theorem we know that the error of this approximation will be of order $O(\Delta t)$. Similarly, for the right-hand term of (B.1) we can write the second order Taylor expansion of $\Delta(\delta \mid x)p(x-\delta,t)$ around $x-\delta = x$ yielding

$$\Delta(\delta \mid x)p(x-\delta,t) \approx \Delta(\delta \mid x)p(x,t) - \delta \cdot \frac{\partial}{\partial x}\left(\Delta(\delta \mid x)p(x,t)\right) + \frac{\delta^2}{2} \cdot \frac{\partial^2}{\partial x^2}\left(\Delta(\delta \mid x)p(x,t)\right). \tag{B.3}$$

Again, by Taylor's theorem we can estimate the error by $O(\delta^2)$. Introducing both approximations in (B.1) and noticing that p no longer depends on δ we can write

$$\begin{aligned} p(x,t)+\Delta t\cdot\frac{\partial p(x,t)}{\partial t} \approx\ & p(x,t)\cdot\underbrace{\int\Delta(\delta\mid x)d\delta}_{1} \\ & -\frac{\partial}{\partial x}\left(p(x,t)\cdot\underbrace{\int\delta\cdot\Delta(\delta\mid x)d\delta}_{E(\Delta)}\right) \\ & +\frac{1}{2}\cdot\frac{\partial^2}{\partial x^2}\left(p(x,t)\cdot\underbrace{\int\delta^2\cdot\Delta(\delta\mid x)d\delta}_{E(\Delta^2)}\right). \end{aligned}$$

As outlined in the equation above, we can identify these integrals with the statistical moments of Δ leading to the well-known diffusion or Kolmogorov forward equation

$$\Delta t\cdot\frac{\partial p(x,t)}{\partial t}\approx-\frac{\partial}{\partial x}\left(p(x,t)\cdot\mathrm{E}[\Delta]\right)+\frac{1}{2}\cdot\frac{\partial}{\partial x^2}\left(p(x,t)\cdot\mathrm{E}\left[\Delta^2\right]\right). \tag{B.4}$$

B.2 Kolmogorov Backward Equation

To derive the Kolmogorov backward equation we follow similar steps as in the previous section. But first, we have to rewrite Equation (B.1) by taking into consideration the starting state of the stochastic process of interest $X(t=0):=x_0$.

$$p(x,t+\Delta t\mid x_0)=\int\Delta(\delta\mid x_0)\cdot p(x,t\mid x_0+\delta)\cdot d\delta. \tag{B.5}$$

Let us consider the left-hand term: after performing a first order Taylor expansion around $\Delta t=0$ we obtain

$$p(x,t+\Delta t\mid x_0)\approx p(x,t\mid x_0)+\Delta t\cdot\frac{\partial p(x,t\mid x_0)}{\partial t}. \tag{B.6}$$

Again, by Taylor's theorem we know that the error of this approximation will be of order $O(\Delta t)$. Similarly, for the right-hand term of (B.5) we can write the second order Taylor expansion of $p(x,t \mid x_0 + \delta)$ around $\delta = 0$ yielding

$$p(x,t \mid x_0 + \delta) \approx p(x,t \mid x_0) + \delta \cdot \frac{\partial p(x,t \mid x_0)}{\partial x_0} + \frac{\delta^2}{2} \cdot \frac{\partial^2 p(x,t \mid x_0)}{\partial x_0^2}. \tag{B.7}$$

Again, by Taylor's theorem we can estimate the error by $O(\delta^2)$. Introducing both approximations in (B.1) and noticing that p no longer depends on δ or we can write

$$\begin{aligned} p(x,t \mid x_0) + \Delta t \cdot \frac{\partial p(x,t \mid x_0)}{\partial t} \approx\ & p(x,t \mid x_0) \cdot \underbrace{\int \Delta(\delta \mid x_0) d\delta}_{1} \\ & + \frac{\partial p(x,t \mid x_0)}{\partial x_0} \underbrace{\int \delta \cdot \Delta(\delta \mid x_0) d\delta}_{E(\Delta)} \\ & + \frac{1}{2} \cdot \frac{\partial^2 p(x,t \mid x_0)}{\partial x_0^2} \underbrace{\int \delta^2 \cdot \Delta(\delta \mid x_0) d\delta}_{E(\Delta^2)}. \end{aligned}$$

As outlined in the equation above, we can identify these integrals with the statistical moments of Δ, leading to the well-known diffusion or Kolmogorov backward equation

$$\Delta t \cdot \frac{\partial p(x,t \mid x_0)}{\partial t} \approx \mathrm{E}[\Delta] \cdot \frac{\partial p(x,t \mid x_0)}{\partial x_0} + \frac{\mathrm{E}\left[\Delta^2\right]}{2} \cdot \frac{\partial p(x,t \mid x_0)}{\partial x_0^2}. \tag{B.8}$$